OPTIMAL PATH GENERATION FOR MONOCULAR SIMULTANEOUS LOCALIZATION AND MAPPING

A Thesis

by

TIMOTHY ISAAC ROORDA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	John E. Hurtado
Committee Members,	John Valasek
	Casey Papovich
Head of Department,	Rodney Bowersox

December 2014

Major Subject: Aerospace Engineering

ABSTRACT

Monocular Simultaneous Localization and Mapping (MonoSLAM), a derivative of Simultaneous Localization and Mapping (SLAM), is a navigation method for autonomous vehicles that uses only an inertial measurement unit and a camera to map the environment and localize the vehicle's position within the environment. Prior to this work, multiple different attempts have been made to generate optimal paths for SLAM, but no optimal path work has been performed specifically for MonoSLAM. The author details an optimal path generation (OPG) method designed specifically for MonoSLAM. In MonoSLAM, the vehicle gains useful data when it can detect a change in bearing to objects in the environment (also known as features). The OPG in question maximizes parallax among all visible features in the environment, with the goal of optimizing fuel usage and estimation accuracy.

In simulations comparing paths from this OPG method with typical MonoSLAM paths, it is evident that the OPG method produces extremely large fuel savings (up to 98%). These fuel savings come at the expense of estimation accuracy, however the OPG method still produces estimation performance that is acceptable for many applications. Looking forward, this work proves that it is indeed possible to improve upon the paths that are typically used in MonoSLAM. This thesis examines a two-dimensional MonoSLAM simulation only; no hardware implementation is performed.

DEDICATION

To Jesus, my Savior.

The waters have not overwhelmed me, and the flames have not consumed me, because of Your great mercy and lovingkindness.

"When you pass through the waters I will be with you; and through the rivers, they shall not overwhelm you; when you walk through fire you shall not be burned, and the flame shall not consume you."

Isaiah 43:2

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. John E. Hurtado, for recruiting me to continue my studies as his graduate student and for supporting me financially and academically. It is rare to find an advisor who cares as much for their students as he does (or who is willing and able to talk about baseball until one of us is late for class), and I am incredibly grateful to have been one of his students.

I would also like to thank Dr. Kevin Brink of the Air Force Research Laboratory for mentoring me during and after my internship at the REEF. His patient help and guidance has helped me to gain a solid foundation and understanding of estimation and Kalman filtering.

I would like to thank Karen Knabe for all of the help she has given me in navigating the details (both small and large) of graduate school. Whether granting me building access, explaining deadlines and procedures, or working on my behalf with the Office of Graduate Studies, Karen has always been a great source of aid. But more importantly, she is a continual source of kindness.

I am incredibly grateful to my family for all of the love and support they have shown me all of my life. Thank you Mom and Dad for committing to putting all of your children through four years of undergraduate schooling out of your own pockets.

Thank you God for always providing for me.

NOMENCLATURE

SLAM	Simultaneous Localization and Mapping
MonoSLAM	Monocular Simultaneous Localization and Mapping
EKF	Extended Kalman Filter
EOMs	Equations of Motion
wrt	with respect to
OCP	Optimal Control Problem
IMU	Inertial Measurement Unit

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Overview

1.1.1 Motivation for SLAM and MonoSLAM

In recent years, GPS navigation has become the navigation method of choice for a large variety of military, commercial, and private applications. GPS navigation is extremely precise, and with the advent of the smart phone, anyone can use the GPS network to navigate. However, a major drawback of GPS is that the GPS signal is relatively weak. This weak signal can easily be lost and even jammed. In most everyday applications (e.g. using one's phone to navigate across town), this is generally not an issue. However, in applications where a vehicle is either routinely operating in a GPS-denied environment or must otherwise be robust to a loss of GPS signal, an alternative method of navigation is required. One such alternative method is Simultaneous Localization and Mapping (SLAM).

SLAM is the process by which a vehicle builds a map of its environment while using that map to concurrently locate itself as it moves through the environment. The SLAM process only utilizes sensors located on the vehicle itself. A common sensor is an Inertial Measurement Unit (IMU), which is composed of accelerometers to measure linear accelerations and gyroscopes to measure angular rates. However, IMUs contain bias and noise on each channel; they provide poor results if one was to integrate the IMU data alone to recover position, velocity, and attitude. Hence, other sensors must be used in conjunction with the IMU. Other possible sensors for navigational use on a vehicle are laser rangefinders, visual cameras, thermal cameras, and magnetic compasses, to name a few.

In Monocular Simultaneous Localization and Mapping (MonoSLAM), the only

sensor complement to the IMU is a single visual camera. This camera is capable of measuring relative bearing from the vehicle to a feature within the environment. (In SLAM, the term feature refers to an object in the environment that the vehicle is calibrated to detect. For instance, if a vehicle is equipped with an infrared camera it could be calibrated to detect infrared strobes.)

1.1.2 Motivation for Optimal Path Generation

In a basic MonoSLAM application, the method only allows the vehicle to localize itself and map the environment given a certain vehicle path; the path is not directly tied to the localization and mapping process, although the SLAM process is directly tied to the path. This thesis is concerned with generating a vehicle path that is dependent upon the vehicle's environment. Namely, the author seeks to answer the question, "How does one generate an *optimal* vehicle path?"

1.2 Literature Review

In the years since the development of the SLAM algorithm, a handful of optimal path generation methods have been developed. Prentice and Roy[1] pioneered a method called the belief roadmap. The belief roadmap seeks to minimize the system covariance (uncertainty) as the vehicle travels from a starting position to a specified final location. This method does not take into account overall path distance.

Makarenko, Williams, Bourgault, and Durrant-Whyte[2] achieve path optimization by considering several weighted factors: utility of information gain, utility of navigation, and utility of localizability. Utility of information gain considers how much new information stands to be gained on a certain path. Utility of navigation seeks to maximize the information gained per distance traveled by negatively weighting longer paths. Utility of localizability takes into account the lowest possible vehicle localization covariance that can be achieved at a spatial point by using only

the features visible at that location. By changing the weight associated with each utility, one can tune the method to provide a path that is taylored to one's objectives.

Kim and Eustice[3] and Stachniss, Grisetti, and Burgard[4] both produced a method which also evaluates a sum of utilities. Their method assigns a reward to paths that maximize information on features already seen while also assigning a reward to paths that enter previously unexplored regions. Kim and Eustice produced a similar method to Stachniss et al, with the main difference being that Kim and Eustice also consider the "plausibility of measurement uncertainty"[3],[4].

Other optimal path generation methods for robotic exploration have been developed, but the methods referenced above are the primary methods that deal directly with SLAM. Davison, Reid, Molton, and Stasse[5] published the first work on SLAM with only a single camera as a sensor complement to the IMU, which they dubbed "Monocular Simultaneous Localization and Mapping" (MonoSLAM). Since their publication in 2007, it appears that no optimal path generation methods have been derived specifically with MonoSLAM in mind.

1.3 Scope of Thesis

The primary purpose of this thesis is to demonstrate proof of concept for a MonoSLAM optimal path generation method. A knowledge of MonoSLAM is required to understand the optimal path generation that is presented, so a detailed overview of MonoSLAM is provided. However, the purpose of the overview is to familiarize the reader with MonoSLAM, not to exhaustively define the algorithm. The reader is encouraged to read Davison et al[5], Sol, Vidal-Calleja, Civera, and Montiel[6], and Brink, Doucette, and Miller[7] for further details of the inner-workings of MonoSLAM.

The work in this thesis concerns a two-dimensional environment in which both

the vehicle and the features are modeled as points (i.e. a land-based vehicle on a flat surface that detects points existing in the same plane). This setup allows for an investigation of the simplest formulation of MonoSLAM that is still sufficiently difficult and provides significant challenges. The work done is simulation only; there is no hardware component.

2. DETAILED OVERVIEW OF MONOSLAM

The simulations performed in this work are meant to emulate a land-based vehicle moving through a flat environment. The vehicle is a modeled as a point moving in a two-dimensional plane that still retains a heading (to maintain knowledge of the orientation of the vehicle's frame with respect to the global frame). The vehicle heading is fixed to always be directed along the vehicle's velocity vector. All environmental features are modeled as points within the two-dimensional plane.

Two main frameworks exist for implementing MonoSLAM: batch estimation and Kalman filtering. This thesis is purely concerned with the Extended Kalman filter implementation.

2.1 The Extended Kalman Filter

The Kalman filter is a predictor-corrector estimator that uses a dynamical system′s governing Equations of Motions (EOMs) combined with measurements to estimate desired system states. The Kalman filter can only be used with systems whose EOMs are linear with respect to the system states and controls, whose measurement equations are linear with respect to the system states, and systems in which all noise is Gaussian. Both the EOMs and the measurement equations in MonoSLAM have nonlinearities that the Kalman filter cannot handle. Thus, the Extended Kalman filter (EKF) must be used. The EKF is a modified Kalman filter that handles nonlinearities by linearizing about a current state estimate. Table 2.1 (which is Table 3.9 from Crassidis and Junkins[8]) and Table 2.2 present the EKF model for a continuous system with discrete measurements.

Table 2.1: Continuous-discrete extended Kalman filter

Model	$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) + G(t)\mathbf{w}(t),\ \mathbf{w}(\mathbf{t}) \sim N(\mathbf{0}, Q(t))$ $\tilde{\mathbf{y}}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k,\ \mathbf{v}_k \sim N(\mathbf{0}, R_k)$
Initialize	$\hat{\mathbf{x}}(t_0) = \hat{\mathbf{x}}_0$ $P_0 = E\left\{\tilde{\mathbf{x}}(t_0)\tilde{\mathbf{x}}^T(t_0)\right\}$
Gain	$K_k = P_k^- H_k^T\left(\hat{\mathbf{x}}_k^-\right)\left[H_k\left(\hat{\mathbf{x}}_k^-\right)P_k^- H_k^T\left(\hat{\mathbf{x}}_k^-\right) + R_k\right]^{-1}$ $H_k\left(\hat{\mathbf{x}}_k^-\right) \equiv \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\big\vert_{\hat{\mathbf{x}}_k^-}$
Update	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k\left[\tilde{\mathbf{y}}_k - \mathbf{h}\left(\hat{\mathbf{x}}_k^-\right)\right]$ $P_k^+ = \left[I - K_k H_k\left(\hat{\mathbf{x}}_k^-\right)\right]P_k^-$
Propagation	$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \mathbf{u}(t), t)$ $\dot{P}(t) = F(t)P(t) + P(t)F^T(t) + G(t)Q(t)G^T(t)$ $F(t) \equiv \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\big\vert_{\hat{\mathbf{x}}(t), \mathbf{u}(t)}$

Table 2.2: Extended Kalman filter variables

Symbol	*Meaning*	*Size*
$\mathbf{x}$	vector of states	$n \times 1$
$\mathbf{u}$	vector of controls	$m \times 1$
$\mathbf{f}$	vector of EOMs	$n \times 1$
$\tilde{\mathbf{y}}$	vector of measurements	$\ell \times 1$
$\mathbf{h}$	vector of measurement equations	$\ell \times 1$
P	matrix of state estimate covariances	$n \times n$
Q	matrix of process noise covariances	$q \times q$
R	matrix of measurement noise covariances	$\ell \times \ell$
K	Kalman gain	$n \times m$
G	matrix by which process noise enters the EOMs	$n \times q$
$\mathbf{w}$	zero-mean Gaussian process-noise	$q \times 1$
$\mathbf{v}$	zero-mean Gaussian measurement noise	$\ell \times 1$

2.2 MonoSLAM-Specific Equations

The MonoSLAM simulations performed in this thesis contain eight vehicle states, three controls, and six process noises. These variables are defined in Table 2.3.

Table 2.3: MonoSLAM vehicle states, controls, and process noises

Variable	*Meaning*
x_1, x_2	global x,y vehicle positions
x_3, x_4	global x, y vehicle velocities
x_5	vehicle heading wrt global $x-$axis
x_6, x_7	IMU x, y acceleration biases
x_8	IMU gyroscope bias
u_1, u_2	vehicle-frame IMU x, y acceleration readings
u_3	IMU gyroscope reading
w_1, w_2, w_3	noise on all three IMU channels
w_4, w_5, w_6	noise defining IMU biases' instantaneous rates of change

2.2.1 Equations of Motion

The governing equations of motion for this system are given as follows:

$$\dot{x}_1 = x_3$$

$$\dot{x}_2 = x_4$$

$$\dot{x}_3 = (u_1 - x_6 + w_1)\cos(x_5) - (u_2 - x_7 + w_2)\sin(x_5)$$

$$\dot{x}_4 = (u_1 - x_6 + w_1)\sin(x_5) + (u_2 - x_7 + w_2)\cos(x_5)$$

$$\dot{x}_5 = u_3 - x_8 + w_3$$

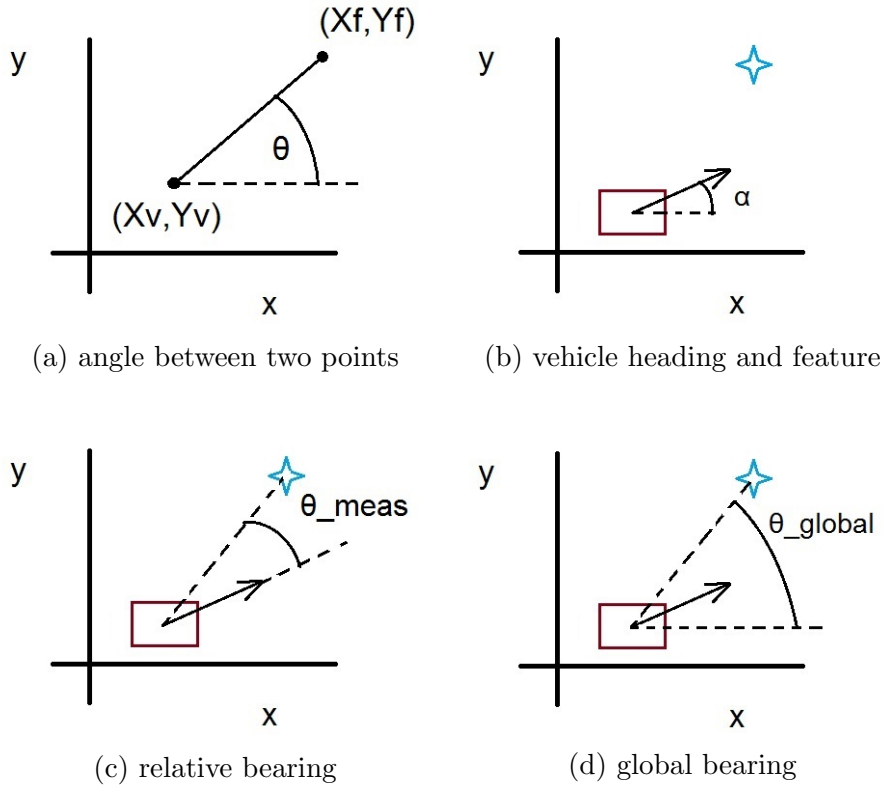
$$\dot{x}_6 = w_4$$

$$\dot{x}_7 = w_5$$

$$\dot{x}_8 = w_6$$

2.2.2 Measurement Equations

There is only one type of measurement available in MonoSLAM: bearing (angle) to a feature in the environment, relative to the vehicle's heading. This is a consequence of using a single camera that is attached to the vehicle. Figure 2.1 represents this measurement visually.



(a) angle between two points (b) vehicle heading and feature

(c) relative bearing (d) global bearing

Figure 2.1: MonoSLAM bearing measurement

From Figures 2.1a and 2.1d, it is evident that the global bearing between two points is calculated with a simple arctangent operation:

$$\theta_{global} = \arctan\left(\frac{y_f - y_v}{x_f - x_v}\right)$$

Figures 2.1b, 2.1c, and 2.1d shed light on the relationship between global bearing, measured (relative) bearing, and vehicle heading:

$$\theta_{meas} = \theta_{global} - \alpha$$

This relationship is expressed in terms of the specified MonoSLAM variables to construct the MonoSLAM measurement equation:

$$h^i(\mathbf{x}) = \arctan\left(\frac{y_f^i - x_2}{x_f^i - x_1}\right) - x_5$$

where i indicates the i-th feature and (x_f^i, y_f^i) refer to the global position of the i-th feature. In the case of known features, x_f^i and y_f^i will simply be known constants. However, the presence of unknown features introduces more complexity into the simulation.

2.3 Accounting for Unknown Features

Unknown features are features that exist in the environment and can be seen by the vehicle camera, however the vehicle has either limited or no information on their true location in the environment. (In this thesis, the vehicle will always have no initial information on unknown features.) Thus, the location of unknown features must be estimated. The parameterization for tracking unknown features that is used in this thesis is known as an anchored modified-polar representation. This parameterization

was first defined by Civera et al[6] and was also discussed by Brink et al[7].

In the anchored modified-polar parameterization, each unknown feature is represented by four states: the two coordinates of the anchor point, global bearing from the anchor point to the unknown feature, and inverse range from the anchor point to the feature. Although this representation is over-parameterized (four variables to define two degrees-of-freedom) and thus introduces more complexity into the system, it significantly decreases nonlinearities in the system (which in turn produces better EKF performance) and allows for immediate feature initialization (by selecting an initial inverse range value with sufficiently large covariance to include a possible infinite range)[6],[7].

The anchored modified-polar parameterization represents an unknown feature with four coordinates:$[x_0, y_0, \theta_0, \rho_0]^T$. The anchor point, (x_0, y_0) is the estimated vehicle position at the instant the unknown feature in question is first seen. The angle θ_0 is the sum of the measured relative bearing to the feature and the estimate of the vehicle heading at the instant the unknown feature is first seen.

The variable ρ_0 is the inverse of the range to the feature from the anchor point. For a detailed discussion about the usage of inverse-range instead of range, the reader is encouraged to examine Civera, Davison, and Montiel[9]. Because an image from a single camera provides no depth perception, no range data exists when the unknown feature is first seen and initialized. For instance, an object in a picture could be a small object that is close to the observer or a large object that is far away from the observer. Thus, ρ_0 is initialized to a value selected by the user; as multiple measurements are taken and feature parallax is induced, ρ_0 will converge to the actual inverse-range value of the unknown feature. Hence, the initially selected value is inconsequential as long as the initial covariance is sufficiently large to cover all possible inverse-range values.

The location of an unknown feature, (x_f, y_f) in terms of $x_0, y_0, \theta_0, \rho_0$ is calculated below:

$$x_f = x_0 + \frac{1}{\rho_0} \cos\theta_0$$

$$y_f = y_0 + \frac{1}{\rho_0} \sin\theta_0$$

2.4 MonoSLAM Environment Test Run

This section includes results from a typical run using the author's MonoSLAM simulation in MATLAB. The simulation runs for 15 seconds with a constant time step of 0.01 seconds; bearing measurements are taken at every time step. The standard deviations used to generate the Gaussian noises that define the process noises, initial state uncertainties, and measurement noises in the system are given in Tables 2.4, 2.5, & 2.6, respectively. These noise values are consistent throughout all simulations performed in this thesis.

Table 2.4: MonoSLAM test run process noises

Process Noise	Standard Deviation	Units
w_1	0.01	$\frac{\text{m}}{\text{s}^2}$
w_2	0.01	$\frac{\text{m}}{\text{s}^2}$
w_3	0.001	$\frac{\text{rad}}{\text{s}}$
w_4	0.01	$\frac{\text{m}}{\text{s}^3}$
w_5	0.01	$\frac{\text{m}}{\text{s}^3}$
w_6	0.001	$\frac{\text{rad}}{\text{s}^2}$

Table 2.5: MonoSLAM test run initial state uncertainties

State	Standard Deviation of Initial Uncertainty	Units
x_1	0.01	m
x_2	0.01	m
x_3	0.01	$\frac{\text{m}}{\text{s}}$
x_4	0.01	$\frac{\text{m}}{\text{s}}$
x_5	0.05	deg
x_6	0.01	$\frac{\text{m}}{\text{s}^2}$
x_7	0.01	$\frac{\text{m}}{\text{s}^2}$
x_8	0.001	$\frac{\text{rad}}{\text{s}}$

Table 2.6: MonoSLAM test run measurement noises

Measurement Noise	Standard Deviation	Units
v_k, $k = 1, 2, ..., N_{features}$	0.05	deg

The vehicle camera has an angular field of view of 180° with a range of 20 meters. The sinusoidal vehicle path used here is a typical MonoSLAM user-programmed vehicle path. The amplitude, period, and phase angle are selected randomly from a user-selected range of values. The vehicle moves from left to right. Figure 2.2 shows the two dimensional vehicle environment, Figures 2.3-2.10 contain the vehicle state error plots, and Figures 2.11-2.14 contain the feature state errors plots.

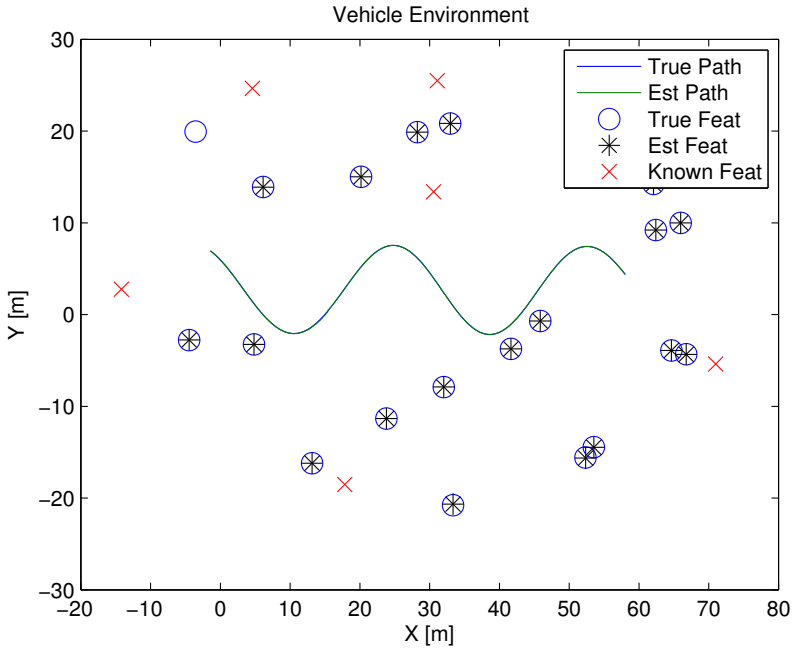


Figure 2.2: Example MonoSLAM vehicle environment

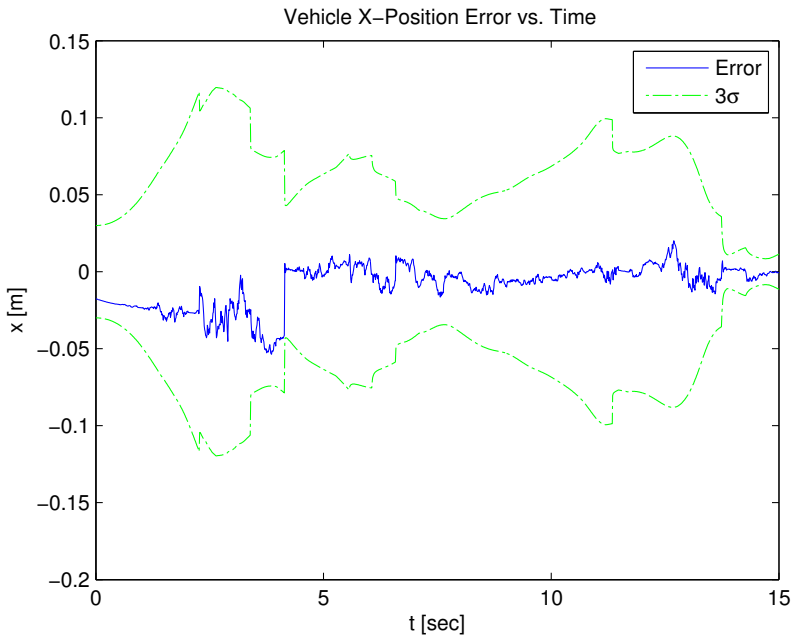


Figure 2.3: Example MonoSLAM vehicle x-position error versus time

Figure 2.4: Example MonoSLAM vehicle y-position error versus time

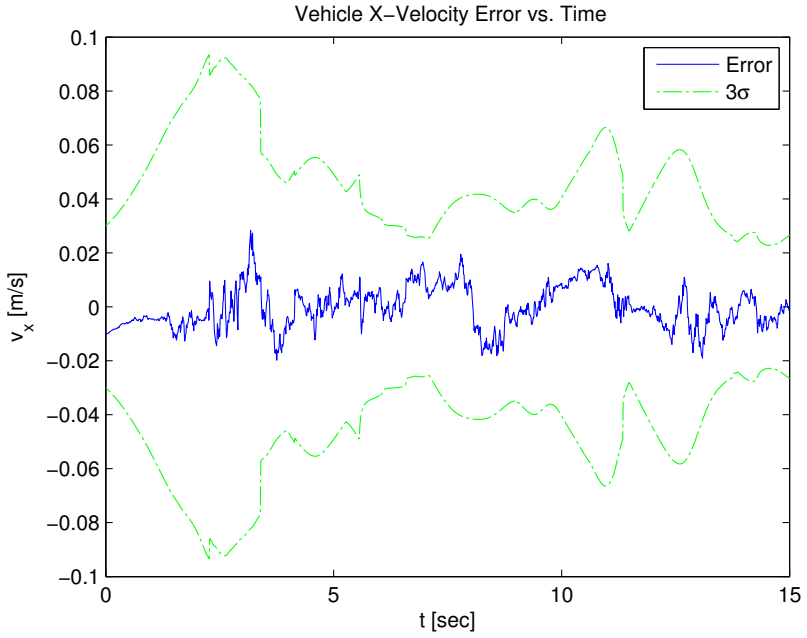


Figure 2.5: Example MonoSLAM vehicle x-velocity error versus time

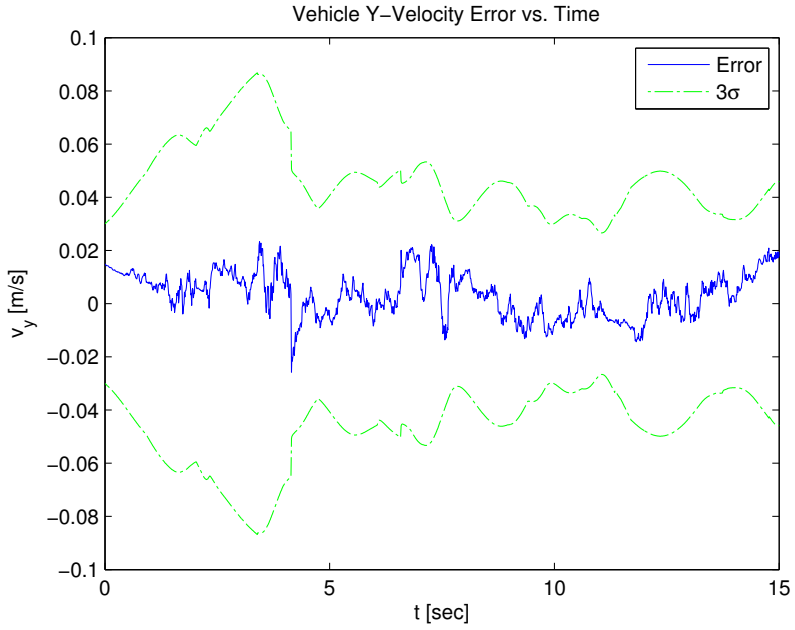


Figure 2.6: Example MonoSLAM vehicle y-velocity error versus time

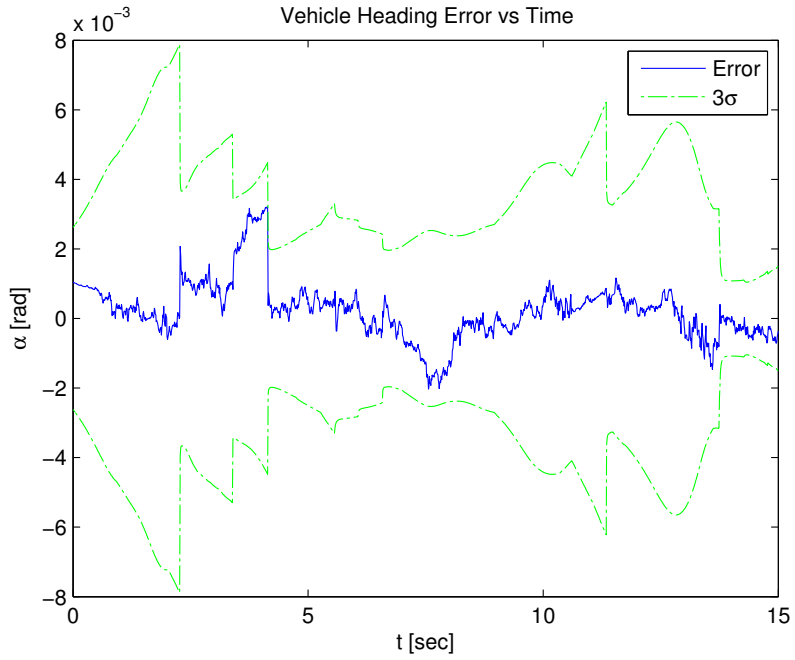


Figure 2.7: Example MonoSLAM vehicle heading error versus time

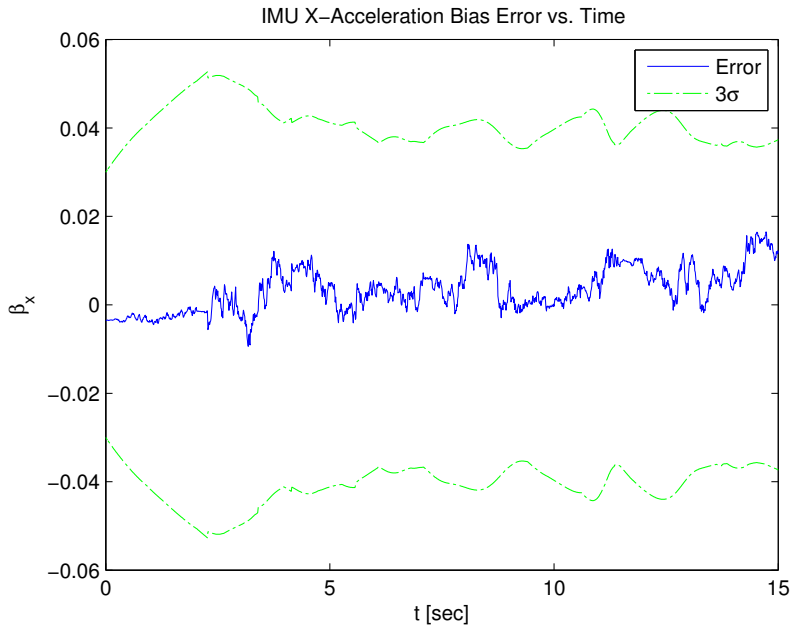


Figure 2.8: Example MonoSLAM IMU x-accelerometer bias error versus time

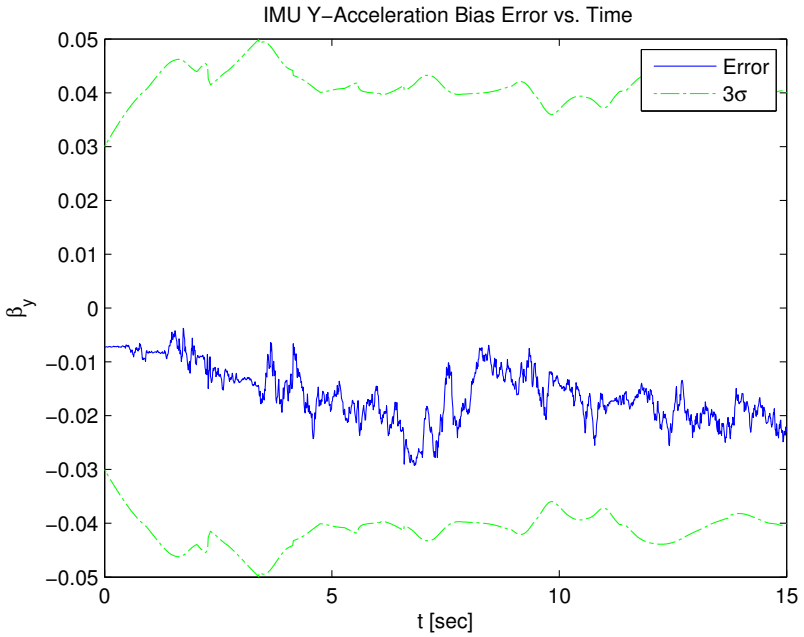


Figure 2.9: Example MonoSLAM IMU y-accelerometer bias error versus time

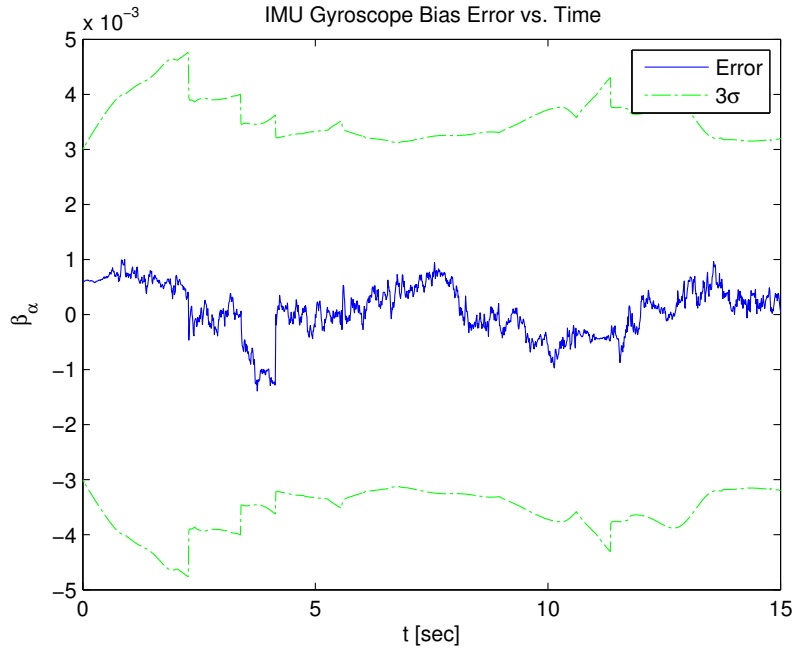


Figure 2.10: Example MonoSLAM IMU gyroscopic bias error versus time

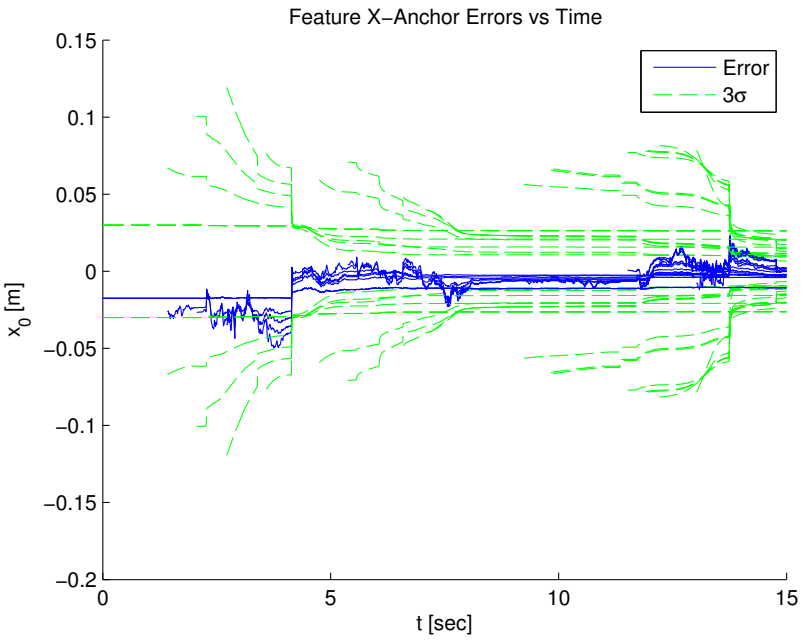


Figure 2.11: Example MonoSLAM unknown feature x-anchor errors versus time

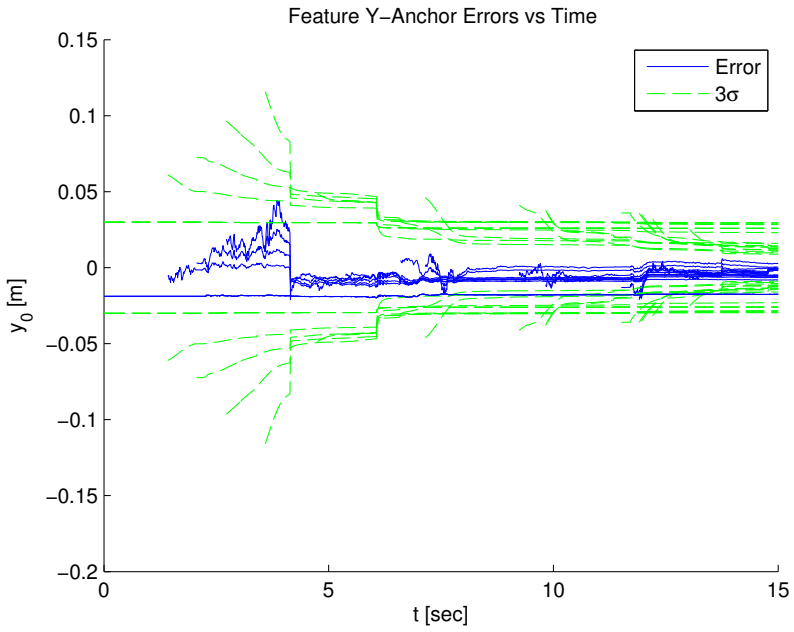


Figure 2.12: Example MonoSLAM unknown feature y-anchor errors versus time

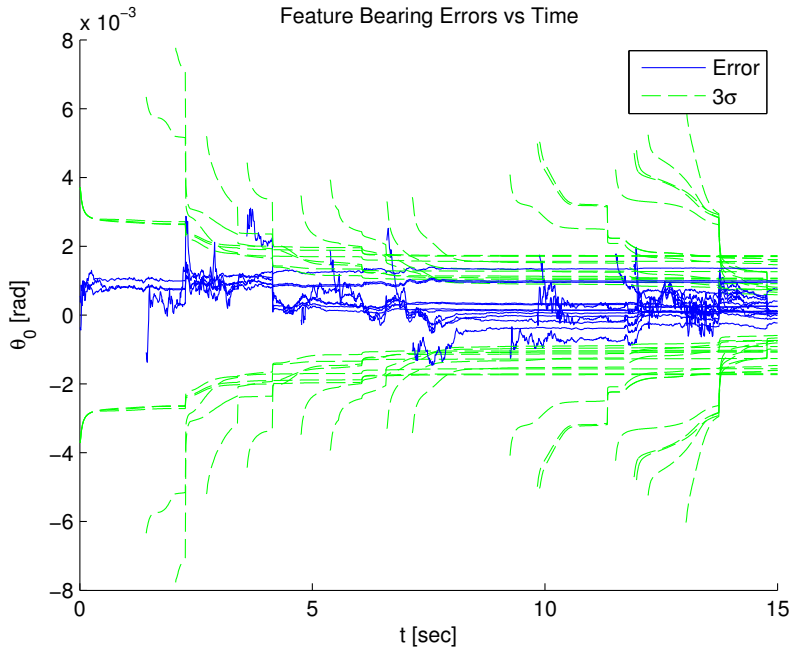


Figure 2.13: Example MonoSLAM unknown feature bearing errors versus time

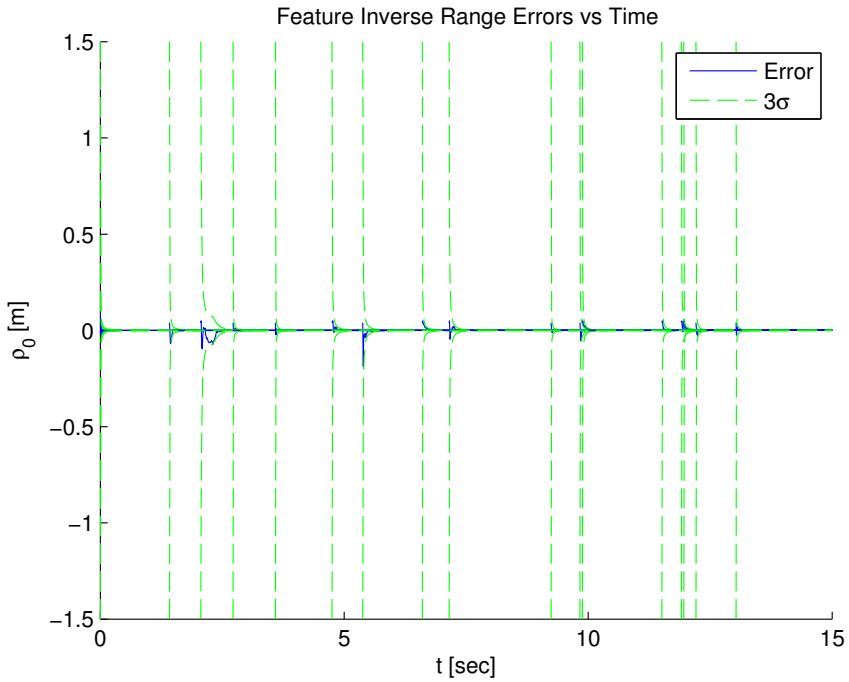


Figure 2.14: Example MonoSLAM unknown feature inverse-range errors versus time

In Figure 2.2, note that the estimated location of the unknown features matches up very closely with their true locations. (One open circle exists in the upper left hand of the plot; this is indicative of an unknown feature that was never seen and initialized by the vehicle.) In the remaining figures, note that the estimation errors lie within their respective 3σ uncertainty boundaries. Both of these observations are indicative of good EKF performance.

3. OPTIMAL PATH GENERATION THEORY

Now that the reader has been introduced to MonoSLAM, the process of optimal path generation will be discussed. Optimal path generation (OPG) will be accomplished in this work by positing and solving an optimal control problem (OCP); see Lewis, Vrabie, and Syrmos[10] for more information about optimal control. There are numerous methods of solving OCPs, but all of them involve a cost function.

A cost function is an expression written in terms of system variables (subject to specified constraints) that one attempts to minimize. For instance, to minimize the fuel used (synonymous with acceleration magnitude) by a vehicle traveling from A to B the following cost function could be used:

$$J = \int_{t_A}^{t_B} \frac{1}{2} \mathbf{a}^T \mathbf{a} \, \mathrm{d}t$$

The author's goal of generating an optimal path for use with MonoSLAM is twofold: to improve estimation performance and fuel usage in comparison with typical MonoSLAM paths. In this section, the author will discuss the composition of the cost function, both qualitatively and quantitatively.

3.1 Feature Parallax

In MonoSLAM, useful information does not come from a single bearing measurement to a feature; but rather it comes from successive bearing measurements to the same feature as the bearing changes over time. This is a phenomenon known as parallax. Put succinctly, greater collective feature parallax provides better estimation performance. Thus, it is desirable to maximize feature parallax over all features (both known and unknown) in order to provide optimal estimation performance in

MonoSLAM.

Change in acceleration and gyroscope readings from the IMU also affect estimation performance; however, to attempt to include these reading fluctuations in a cost function would add significant and unnecessary complexity. Additionally, maximizing changes in IMU readings directly conflicts with one of the purposes of optimal path generation: to limit fuel costs. It is the intent of the author to focus solely on maximizing feature parallax with the hope that the optimal path generated by doing so will provide sufficient IMU readings for acceptable state estimation while also providing lower acceleration magnitudes than typical MonoSLAM paths.

3.2 OPG Cost Function Formulation

The mathematical expression for in-line travel between a vehicle and a feature is

$$\mathbf{v} \cdot (\mathbf{r}_f - \mathbf{r})$$

where $\mathbf{v}$ is the global vehicle velocity vector, $\mathbf{r}$ is the global vehicle position vector, and $\mathbf{r}_f$ is the global feature position vector. Maximizing parallax corresponds to minimizing the magnitude of in-line travel distance, which is expressed as

$$\frac{1}{2}\left[\mathbf{v} \cdot (\mathbf{r}_f - \mathbf{r})\right]^2$$

(the $\frac{1}{2}$ is used for convenience). The above expression applies to a single feature. Sum this expression across all features in order to obtain the magnitude of the collective in-line travel distance.

$$\frac{1}{2}\sum_{i=1}^{N_f}\left\{\left[\mathbf{v} \cdot \left(\mathbf{r}_f^i - \mathbf{r}\right)\right]^2\right\}$$

where N_f is the total number of features and $\mathbf{r}_f^i$ is the global position vector of the i-th feature. Integrate over time in order to complete the cost function.

$$J = \frac{1}{2} \int \sum_{i=1}^{N_f} \left\{ \left[\mathbf{v} \cdot \left(\mathbf{r}_f^i - \mathbf{r} \right) \right]^2 \right\} \mathrm{d}t$$

Coordinatized in the two-dimensional Cartesian reference frame and expressed in scalars only, the cost function is

$$J = \frac{1}{2} \int \sum_{i=1}^{N_f} \left\{ \left[v_x \left(x_f^i - r_x \right) + v_y \left(y_f^i - r_y \right) \right]^2 \right\} \mathrm{d}t$$

3.3 Optimal Control Problem Statement

For the purposes of this thesis, the path generation OCP will be an initial and final time-fixed problem with specified initial and final vehicle positions. The final formulation of this problem is as follows:

$$\min J = \frac{1}{2} \int_{t_0}^{t_f} \sum_{i=1}^{N_f} \left\{ \left[v_x \left(x_f^i - r_x \right) + v_y \left(y_f^i - r_y \right) \right]^2 \right\} \mathrm{d}t$$

subject to

$$\dot{x} = v_x \qquad \dot{y} = v_y$$

$$x\left(t_0\right) = x_0 \qquad y\left(t_0\right) = y_0$$

$$x\left(t_f\right) = x_f \qquad y\left(t_f\right) = y_f$$

$$t_0 = \text{given} \qquad t_f = \text{given}$$

3.4 Integrating Optimal Path Generation with MonoSLAM

Now that an optimal control problem is formulated, one must consider how to properly integrate the optimal path generation scheme with the MonoSLAM algorithm. First, the continuous-time problem is discretized using a 1st order Euler method.

$$\min J = \frac{1}{2}\sum_{k=1}^{N-1}\sum_{i=1}^{N_f}\left\{\left[v_{x_k}\left(x_f^i - r_{x_k}\right) + v_{y_k}\left(y_f^i - r_{y_k}\right)\right]^2\right\}\Delta t_k$$

subject to

$$x_{k+1} = x_k + v_{x_k}\Delta t_k,\ k = 1, 2, ..., N-1$$

$$y_{k+1} = y_k + v_{y_k}\Delta t_k,\ k = 1, 2, ..., N-1$$

$$t_{k+1} = t_k + \Delta t_k,\ k = 1, 2, ..., N-1$$

$$x_1 = x_0 \qquad y_1 = y_0$$

$$x_N = x_f \qquad y_N = y_f$$

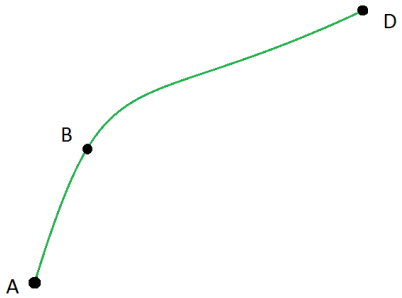
$$t_1 = t_0 \qquad t_N = t_f$$

Second, the OCP is solved using the "fmincon" MATLAB routine with a cost function that includes all features that are seen by the vehicle at the starting time step. At this point, a complete trajectory exists that collectively maximizes parallax over all features that were seen initially. However, unless all features in the environment are seen at every time step, this initial trajectory is not representative of a true maximum parallax. In order to assure this, an iterative solution process is added. A user-selected parameter dictates how many steps from the original trajectory solution are kept and added to the desired trajectory. Then, starting at the indicated time

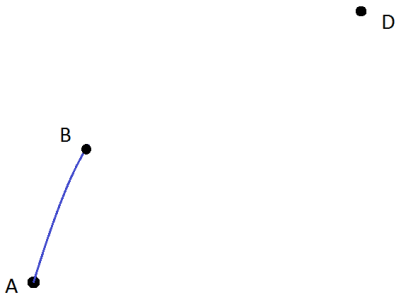
step, a new cost function is formulated using all features that are seen at the current step and the corresponding OCP is formulated and solved. This process is iterated until the vehicle arrives at the specified final position. This process is visualized in Figure 3.1.

Figure 3.1 demonstrates the iterative solution process in the absence of estimation. In reality, the vehicle will not know it's true starting position for each OCP formulation. Instead, the vehicle will have its best estimate of its current position and use that data to plot a trajectory from where it thinks it is to the desired finishing position. The vehicle calculates an optimal trajectory from A to B and then implements the MonoSLAM algorithm as it moves along the trajectory. Thus, optimal path generation and MonoSLAM work hand-in-hand.

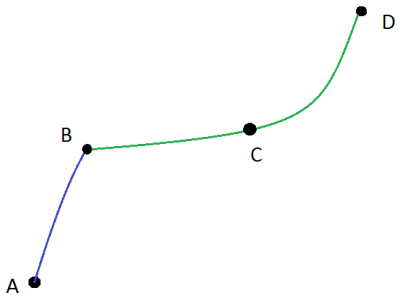
(a) Desired start and finish



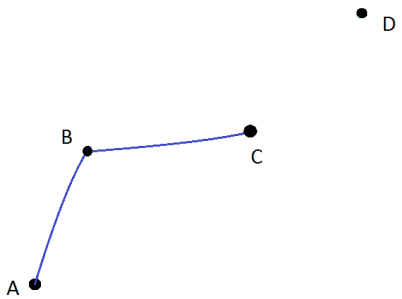
(b) Solution from A to D



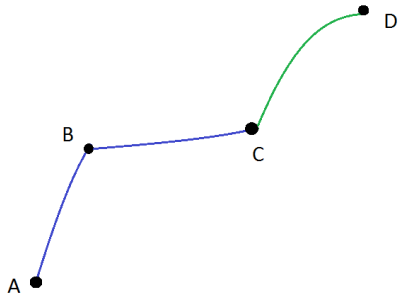
(c) AB added to trajectory



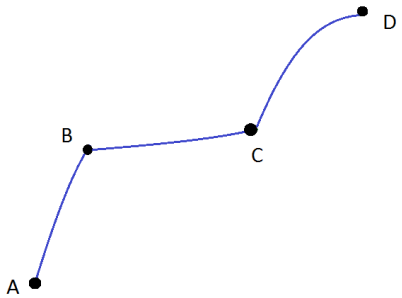
(d) Solution from B to D



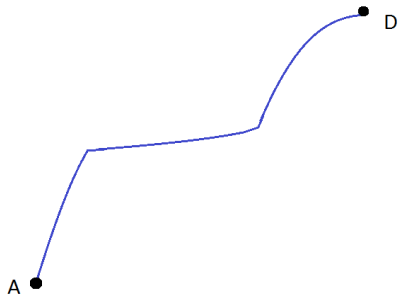
(e) BC added to trajectory



(f) Solution from C to D



(g) CD added to trajectory



(h) Complete iterated trajectory

Figure 3.1: Iterative optimal path generation solution process

4. SIMPLE OPTIMAL PATH GENERATION EXAMPLES

Now that an optimal control problem has been formulated, optimal path generation solutions can be found. This section will cover several simple examples, starting with the simplest and building in complexity. (All of the features in this section will be known features; unknown features present additional challenges for cost function-inclusion, which will be discussed in Section 5.)

4.1 Environment with One Known Feature

The simplest problem to begin with is a vehicle navigating from a specified starting position to a specified ending position in an environment containing only one known feature that is seen by the camera at every instant in time. For such an environment, the problem is more tractable when expressed in a polar frame that is centered on the feature. The vehicle position and velocity vectors are written in this frame as

$$\mathbf{r} = r\hat{\mathbf{e}}_{\mathbf{r}}$$

$$\mathbf{v} = v_r\hat{\mathbf{e}}_{\mathbf{r}} + v_\theta\hat{\mathbf{e}}_\theta = \dot{r}\hat{\mathbf{e}}_{\mathbf{r}} + r\dot{\theta}\hat{\mathbf{e}}_\theta$$

With this coordinate change, the OCP is rewritten as:

$$\min J = \int_{t_0}^{t_f} \frac{1}{2} v_r^2 r^2 dt$$

subject to

$$\dot{r} = v_r \qquad \dot{\theta} = \frac{v_\theta}{r}$$

$$r(t_0) = r_0 \qquad r(t_f) = r_f$$

$$\theta(t_0) = \theta_0 \qquad \theta(t_f) = \theta_f$$

$$t_0 = \text{given} \qquad t_f = \text{given}$$

Note that only radial velocity and vehicle radius are included in the cost function; transverse movement does not have an associated cost. This aligns with the author′s choice of cost function: radial (in-line) travel provides no parallax.

The simplicity of this problem allows for a well-defined analytical solution. For comparison, a numerical solution is also provided. This section is only concerned with generating an optimal path for use with MonoSLAM. The MonoSLAM algorithm itself is not applied to this example because an environment with only a single feature lends itself to very poor estimation performance.

4.1.1 Analytical Solution

The analytical solution technique presented in this section is an indirect method requiring formulation of the Hamiltonian and application of the necessary optimality conditions. (For more information about this solution technique and other OCP solutions, the reader is encouraged to refer to Lewis et al[10]).

Form the Hamiltonian and derive the necessary conditions:

$$H = \frac{1}{2}v_r^2 r^2 + \lambda_r v_r + \lambda_\theta \frac{v_\theta}{r}$$

$$\frac{\partial H}{\partial \lambda_r} = \dot{r} = v_r$$

$$\frac{\partial H}{\partial \lambda_\theta} = \dot{\theta} = \frac{v_\theta}{r}$$

$$-\frac{\partial H}{\partial r} = \dot{\lambda}_r = -v_r^2 r + \lambda_\theta \frac{v_\theta}{r^2}$$

$$-\frac{\partial H}{\partial \theta} = \dot{\lambda}_\theta = 0$$

$$\frac{\partial H}{\partial v_r} = 0 = v_r r^2 + \lambda_r$$

$$\frac{\partial H}{\partial v_\theta} = 0 = \frac{\lambda_\theta}{r}$$

Combining the above equations gives the following results:

$$\ddot{r}r^2 + \dot{r}^2 r = 0$$

$$\lambda_\theta = 0$$

Because the θ costate is equal to zero, the OCP puts no constraints on θ motion (as was discussed earlier in this section). All that is left is to solve the differential equation in r.

Factor out the common r in both terms:

$$r\left(\ddot{r}r + \dot{r}^2\right) = 0$$

The author assumes that r can never be 0, and thus the only remaining solution satisfies

$$\ddot{r}r + \dot{r^2} = 0$$

This is a second order, nonlinear, ordinary differential equation that has an analytical solution (it is a perfect differential). The solution is straightforward and is therefore left to the reader as an exercise. After applying the specified boundary conditions, the solution is

$$r(t) = \sqrt{\frac{r_0^2(t_f - t) + r_f^2(t - t_0)}{t_f - t_0}}$$

4.1.2 Numerical Solution

As this example has an analytical solution, it is insightful to also solve the problem numerically using the same discretization and solution techniques that will be used throughout this work. Applying the discretization technique from Section 3.4 to the single-feature OCP yields the following discrete time OCP:

$$\min J = \sum_{k=1}^{N-1} \frac{1}{2} v_{r_k}^2 r_k^2 \Delta t_k$$

subject to

$$r_{k+1} = r_k + v_{r_k}\Delta t_k,\ k = 1, 2, ..., N-1$$

$$\theta_{k+1} = \theta_k + \frac{v_{\theta_k}}{r_k}\Delta t_k,\ k = 1, 2, ..., N-1$$

$$r_1 = r_0 \qquad \theta_1 = \theta_0$$

$$r_N = r_f \qquad \theta_N = \theta_f$$

This problem is solved using the fmincon MATLAB function. The results are given in Section 4.1.3.

4.1.3 Comparison

Figures 4.1-4.4 show the analytical and discrete optimal solutions for a system with $t_0 = 0$ seconds, $t_f = 5$ seconds, $r_0 = 3$ meters, $r_f = 3$ meters, $\theta_0 = \frac{\pi}{4}$ radians, $\theta_f = \frac{3\pi}{4}$ radians, and a constant discrete time step of $\Delta t = 0.1$ seconds. The polar component of the analytical solution was selected such that $\dot{\theta} = \text{constant}$.

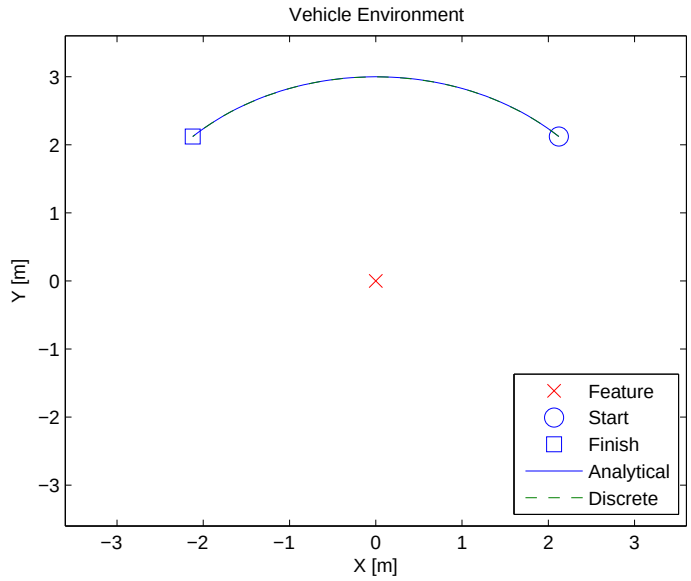


Figure 4.1: Identical radii single-feature OCP vehicle environment

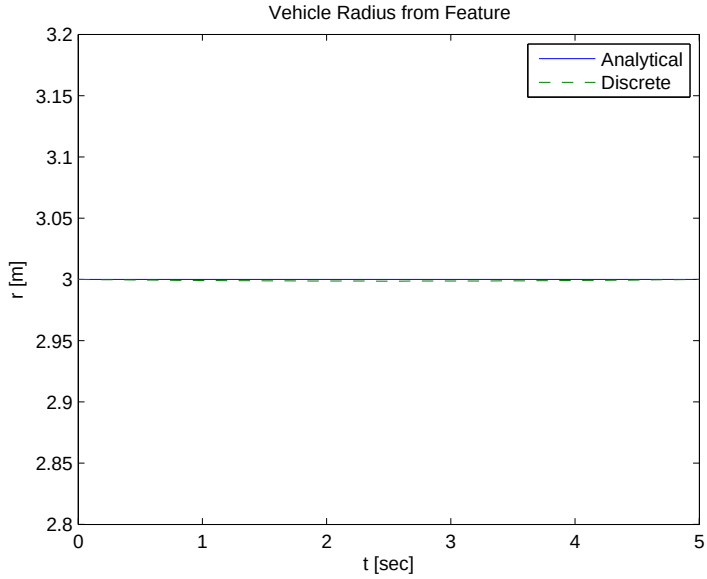


Figure 4.2: Identical radii single-feature OCP radius versus time

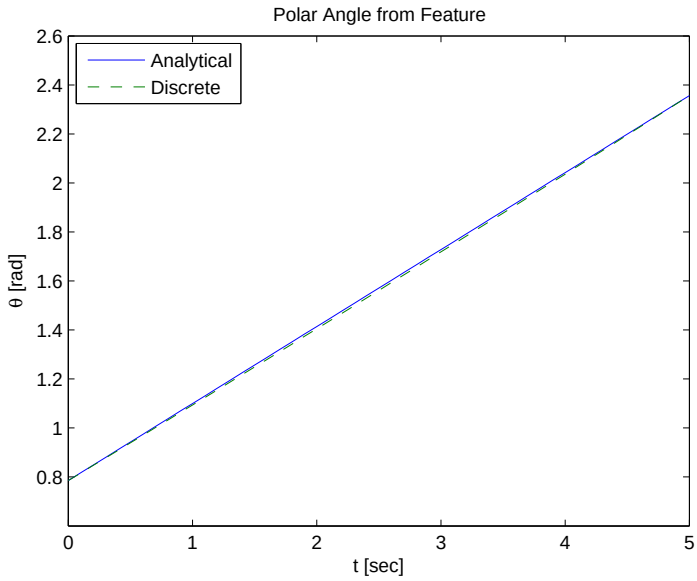


Figure 4.3: Identical radii single-feature OCP polar angle versus time

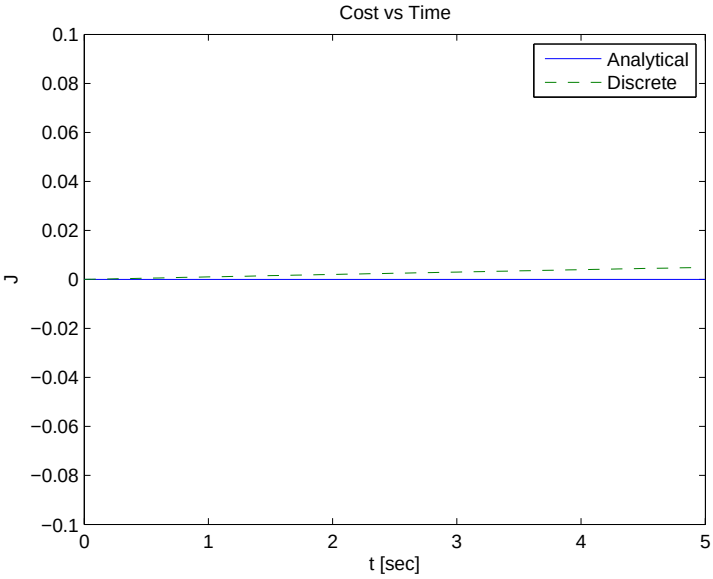


Figure 4.4: Identical radii single-feature OCP cost versus time

The optimal path lies along the arc of radius $r = 3$. Note in Figure 4.4 that no cost is induced because the start radius and finish radius are equivalent. The discrete solution very closely approximates the analytical results; any deviations that exist will dissipate as the discrete time step becomes infinitesimally small.

Now consider a single-feature environment with different beginning and ending radii. Figures 4.5-4.8 show the analyical and discrete optimal solutions for a system with $t_0 = 0$ seconds, $t_f = 5$ seconds, $r_0 = 3$ meters, $r_f = 1$ meter, $\theta_0 = \frac{\pi}{4}$ radians, $\theta_f = \pi$ radians, and a constant discrete time step of $\Delta t = 0.1$ seconds. The polar component of the analytical solution was again selected such that $\dot{\theta} = \text{constant}$.

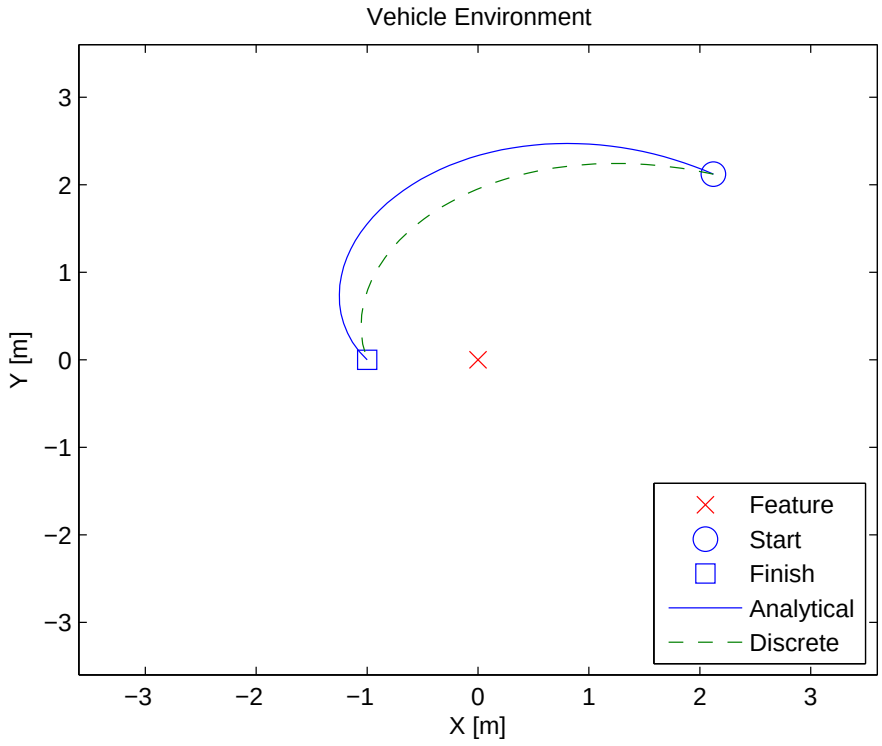


Figure 4.5: Different radii single-feature OCP vehicle environment

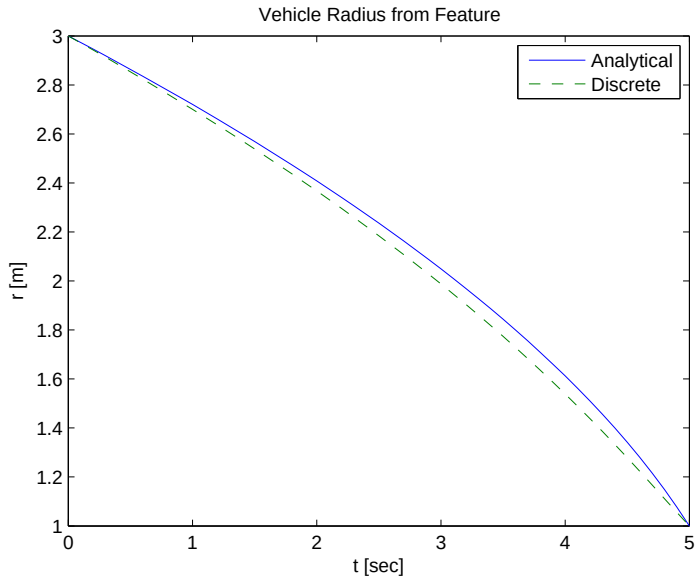


Figure 4.6: Different radii single-feature OCP radius versus time

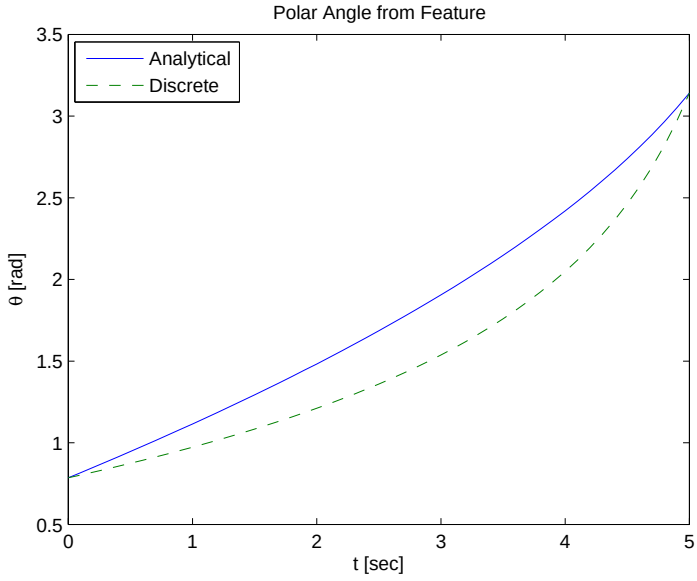


Figure 4.7: Different radii single-feature OCP polar angle versus time

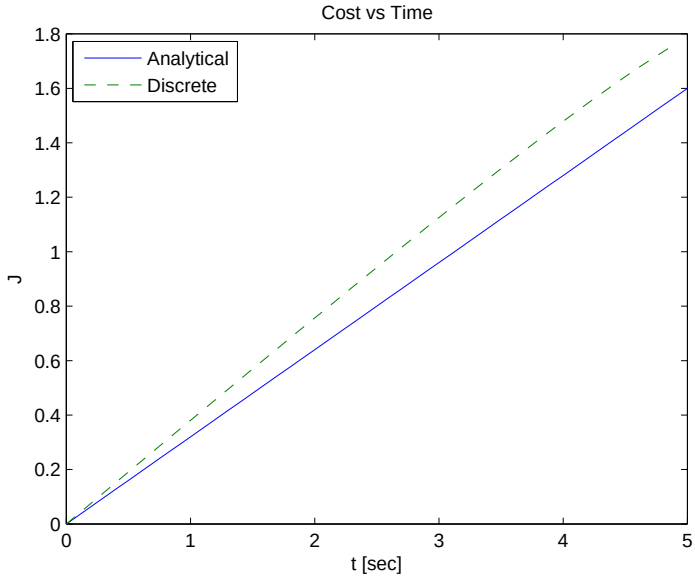


Figure 4.8: Different radii single-feature OCP cost versus time

The results show that the optimal path between two points of different radii is a spiral. Note in Figure 4.6 that the radius versus time plot is almost identical for both solutions. The deviations between these two radial solutions come from discretization error. These small deviations are magnified when evaluating the cost function (in Figure 4.8 the total analytical cost is 1.6, while the total discrete cost is approximately 1.8). It must also be noted that while both of the radius versus time solutions are close to each other, the polar angle versus time solutions in Figure 4.7 are noticeably different. This is because the discrete OCP formulation placed no constraints on θ-motion, whereas the analytical OCP formulation required that $\dot{\theta}$ be a constant. This difference in θ-motion causes the analytical path and the discrete path to look significantly different.

Both of the examples solved in this section serve to give an initial insight into the shape of optimal paths, confirm that the author's choice of cost function achieves

the desired optimality, and affirm that the author's discrete solution technique works properly.

4.2 Line of Known Features

In this section, the author will examine optimal path generation combined with MonoSLAM for an environment in which all of the features are known features that lie on the line connecting the specified vehicle start and finish positions. In this example, no limitation is placed upon camera visibility; all known features are always seen by the vehicle camera. The simulation runs for 2 seconds with a time step of 0.05 seconds. This environment is shown in Figure 4.9

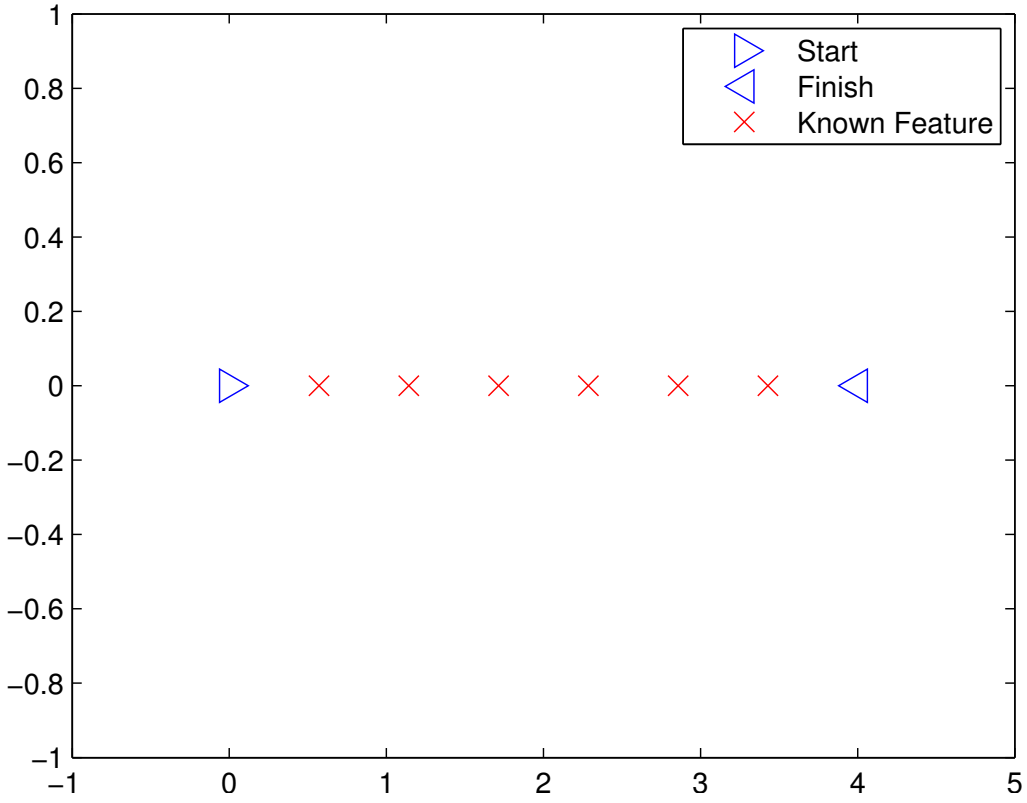


Figure 4.9: Line of known features

Applying the integrated OPG MonoSLAM scheme as discussed in Section 3.4 yields the true and estimated vehicle paths shown in Figure 4.10 and the vehicle state error plots given in Figures 4.11-4.18.

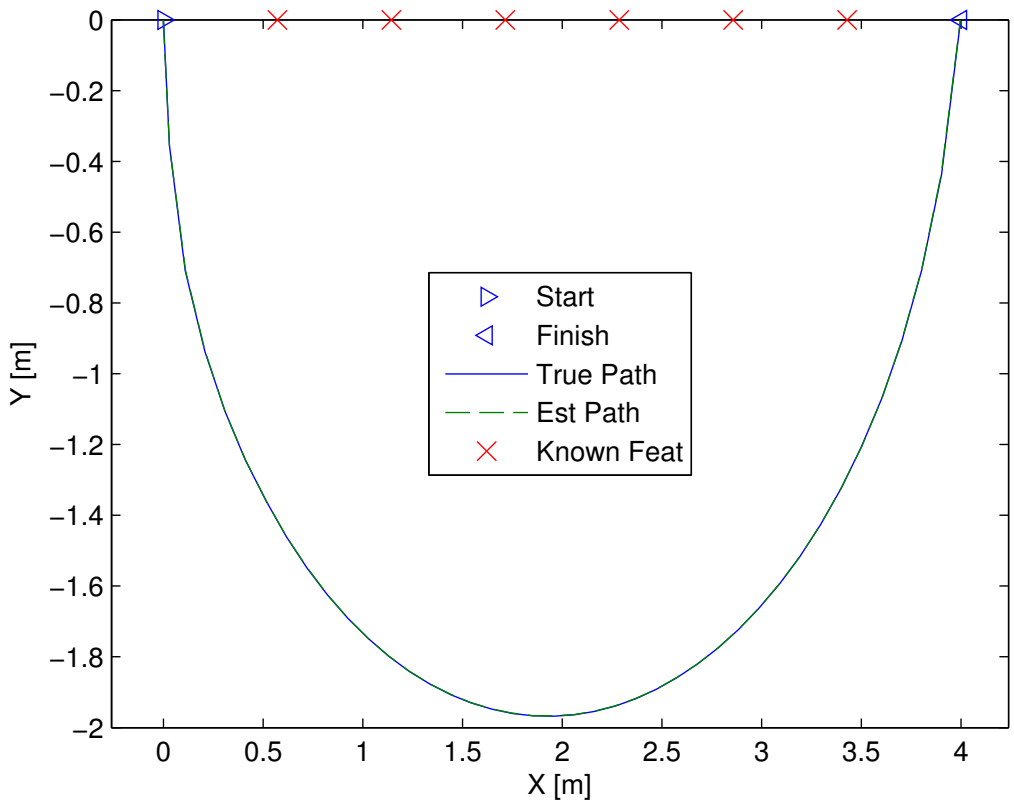


Figure 4.10: Line of known features vehicle environment

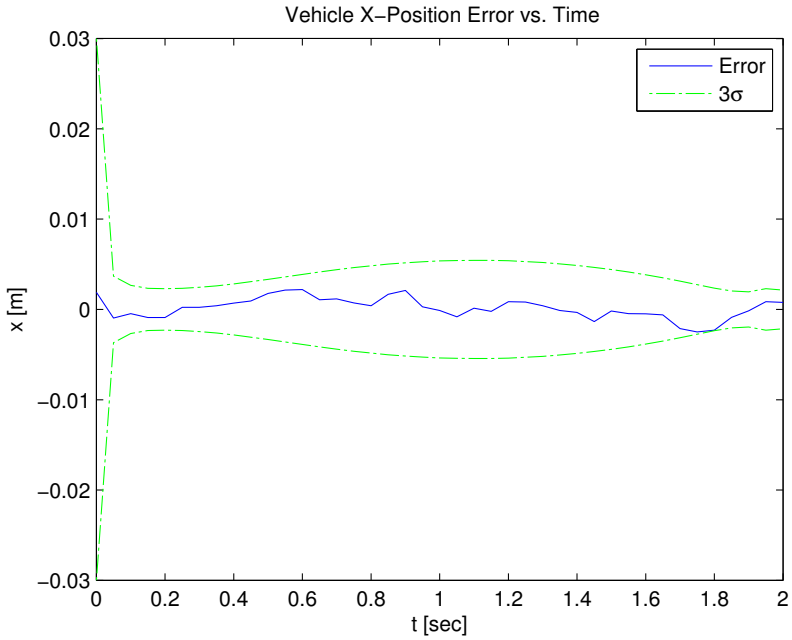


Figure 4.11: Line of known features vehicle x-position error versus time

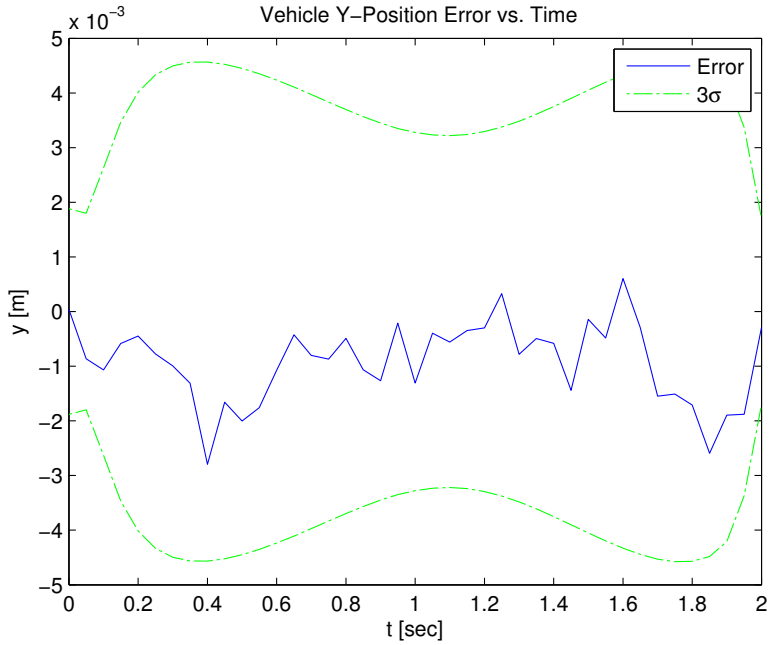


Figure 4.12: Line of known features vehicle y-position error versus time

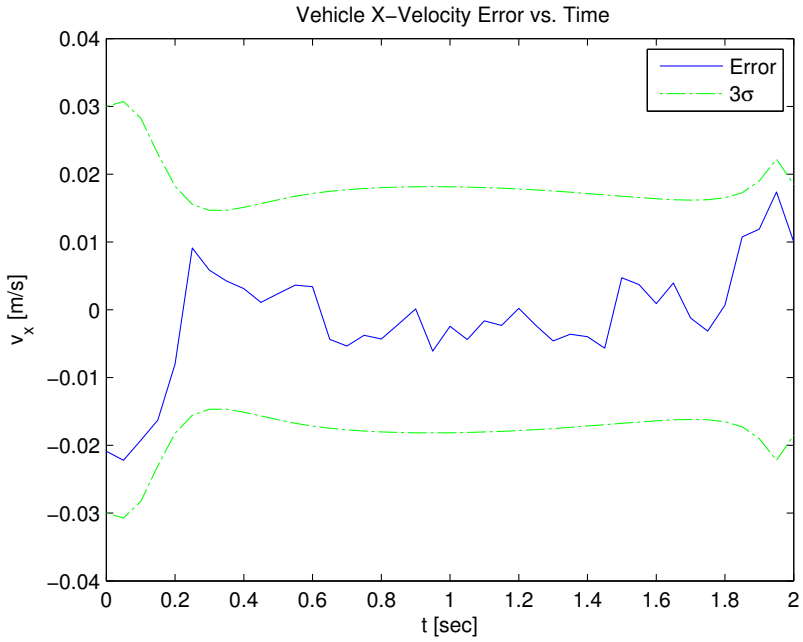


Figure 4.13: Line of known features vehicle x-velocity error versus time

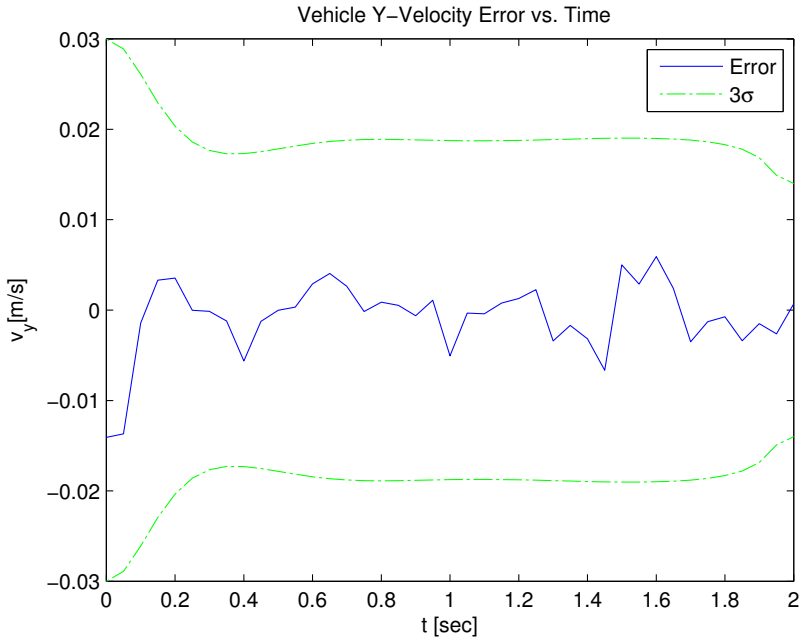


Figure 4.14: Line of known features vehicle y-velocity error versus time

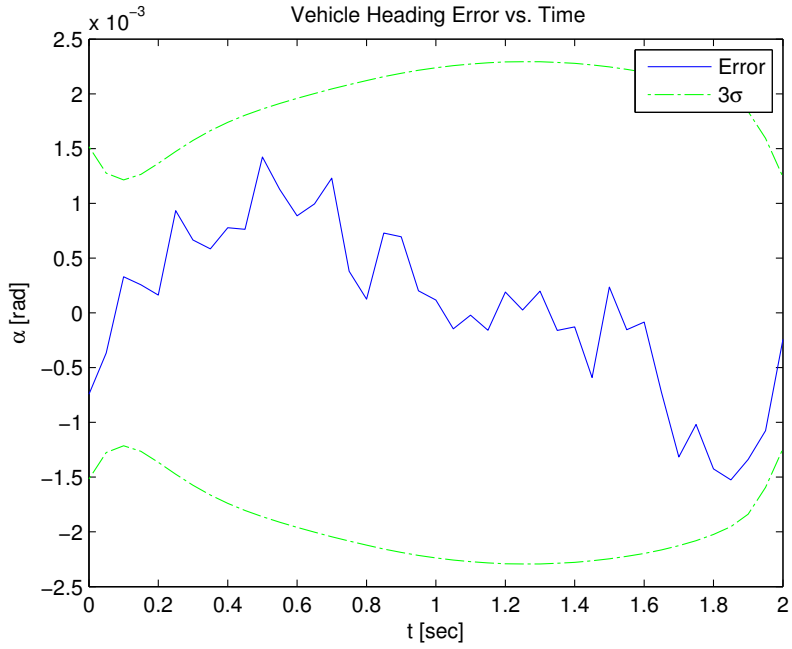


Figure 4.15: Line of known features vehicle heading error versus time

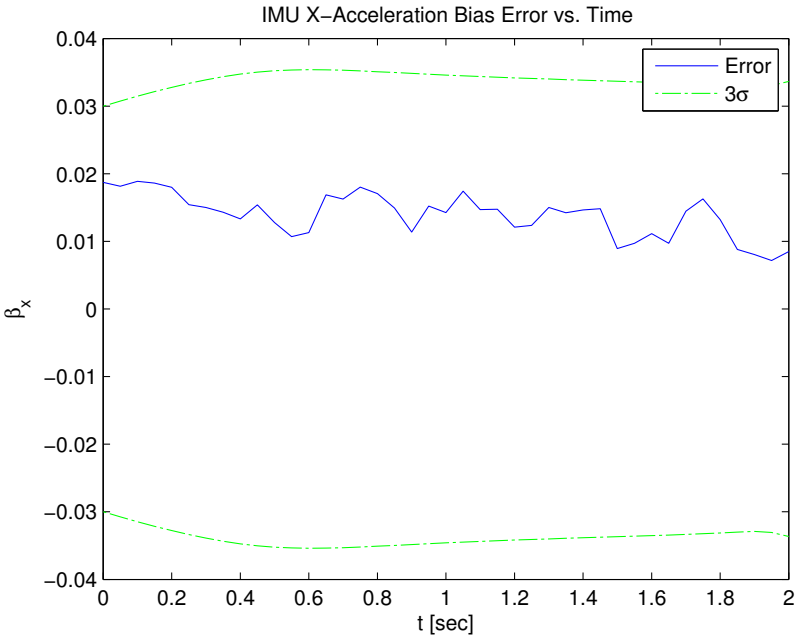


Figure 4.16: Line of known features IMU x-accelerometer bias error versus time

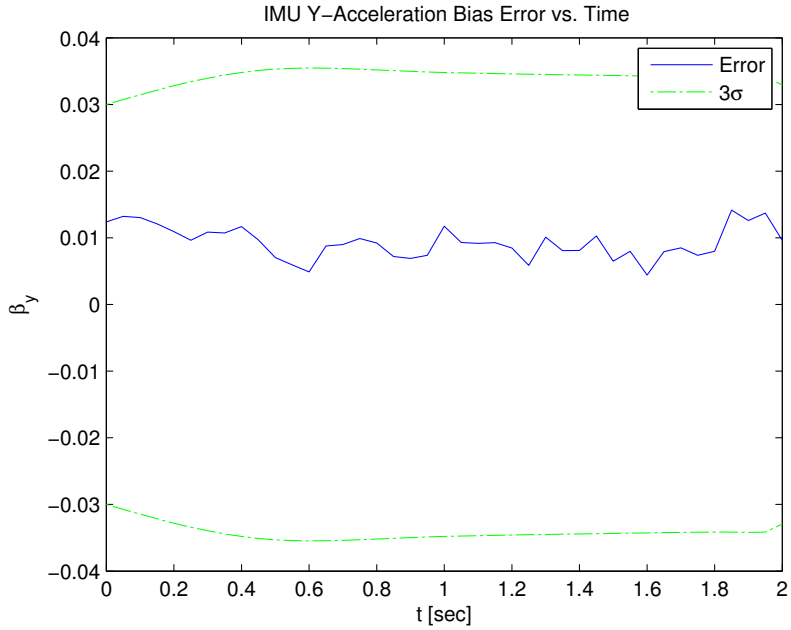


Figure 4.17: Line of known features IMU y-accelerometer bias error versus time

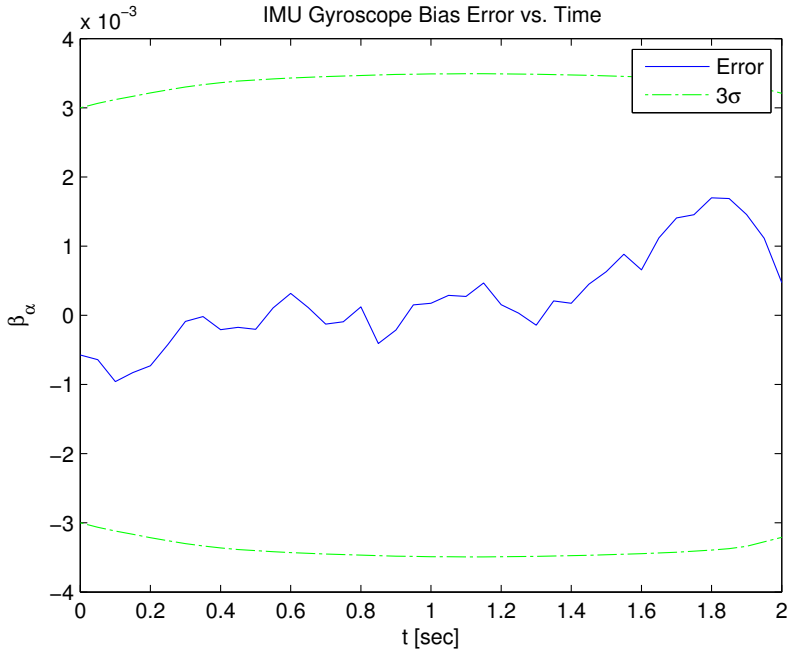


Figure 4.18: Line of known features IMU gyroscopic bias error versus time

The path generated in Figure 4.10 is a parabola. The parabolic trajectory takes the vehicle to its apex where collective feature parallax is at a maximum. Because of the final position constraint, the vehicle must leave this apex and reach the specified ending position. Additionally, all eight of the EKF state estimates are well within their respective 3σ bounds, indicating good estimation performance by MonoSLAM. This example illustrates that the optimal path generation performs as desired and works well in tandem with the MonoSLAM algorithm.

5. OPTIMAL PATH GENERATION WITH UNKNOWN FEATURES

Section 4 examined optimal path generation for MonoSLAM while considering only known features in the environment. In this section, the challenge and results of including unknown features into the optimal path generation scheme will be covered.

5.1 Methodology for Cost Function Inclusion

The cost function developed in Section 3.2 includes the position of features within the environment.

$$J = \frac{1}{2} \int \sum_{i=1}^{N_f} \left\{ \left[v_x \left(x_f^i - r_x \right) + v_y \left(y_f^i - r_y \right) \right]^2 \right\} \mathrm{d}t$$

It is straightforward to include a known feature in the cost function, as its position is (by the very definition of a known feature) known. However, the inclusion of unknown features in the cost function is not as straightforward.

Feature cost function inclusion can be likened to an $'$on/off$'$ switch. Known Features are switched $'$on$'$ and included in the cost function if they are seen at the instant the OCP is formulated. Unknown features are switched $'$on$'$ when they (a) are visible at the instant the OCP is formulated, (b) have been seen a certain number (specified by the user) of times previously, and (c) the vehicle has estimated the unknown feature's position sufficiently well. An unknown feature's position is considered to be estimated sufficiently well when the most recent change in position estimate is less than a change in position parameter value that is selected by the user. Mathematically, this is expressed as

$$\sqrt{\left(\Delta x_f\right)^2 + \left(\Delta y_f\right)^2} \leq \sigma_f$$

where Δx_f and Δy_f are the most recent changes in estimated x-position and y-position, respectively, and σ_f is the user-selected parameter value. Generally, the largest change in estimated unknown feature position occurs between the first few times an unknown feature is seen. The first time an unknown feature is seen, its inverse depth is initialized at a very low value (because no depth information is available), which means that the estimated feature position is usually a sizable distance away from the truth. The next few times that feature is seen, the feature parallax allows the vehicle to localize the feature much closer to the truth than the original estimate. After these first few estimation steps, any change in the estimated unknown feature position is small, and (as long as σ_f is not too small) the unknown feature is eligible for cost function inclusion. The constraint on how many times the feature has been seen also works to ensure the feature has been localized sufficiently well to include in the cost function.

It should be noted that optimal path generation's success does not depend significantly on how close an unknown feature's position estimate is to the truth. If an unknown feature is included in the cost function with a poor location estimate, it is only one feature out of (possibly) many included in the cost function. If every feature included in the cost function is an unknown feature with a poor position estimate, it is likely that the optimal path generated will provide sufficient parallax with respect to the true feature positions, even if perhaps not maximum parallax. Additionally, as the vehicle moves through the environment, MonoSLAM will produce better unknown feature location estimates.

This section will examine the performance of MonoSLAM optimal path generation for an environment containing only unknown features. Figures 5.1-5.13 are representative of somewhat acceptable estimation performance within an all-unknown feature environment; Figures 5.14-5.26 are representative of very poor estimation

performance within an all-unknown feature environment.

The results are generated by running the same simulation twice. The simulation runs for 2 seconds with a timestep of 0.1 seconds. The vehicle camera has a field of view of 180° and extends out to 100 meters. The unknown feature change in estimated position parameter, σ_f, is 10 meters, and unknown features must be seen 4 times previously in order to be eligible for cost function inclusion. The cost function is reformulated every 4 steps.

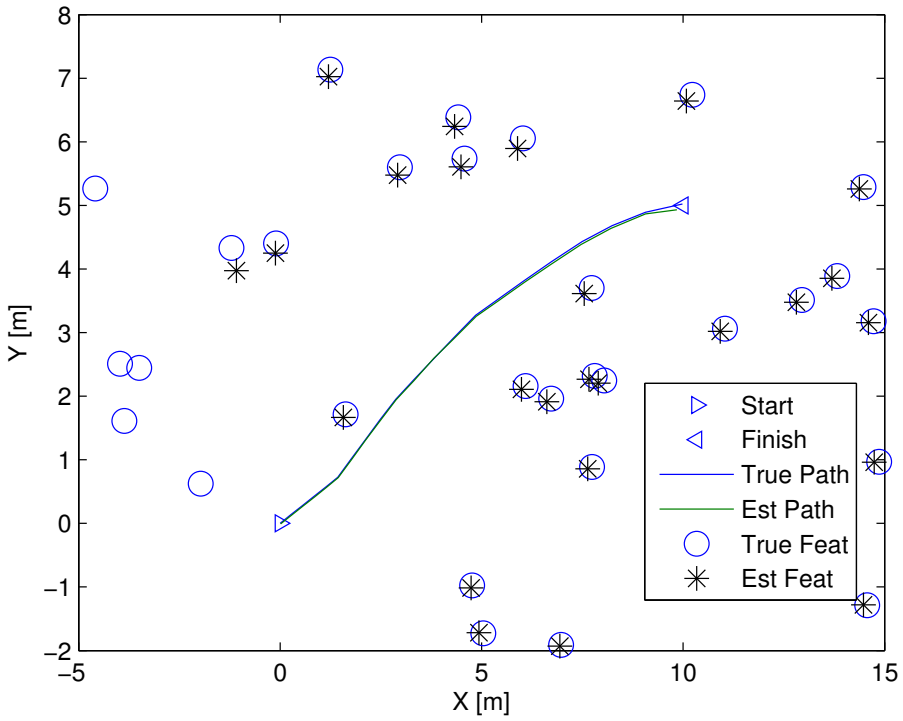


Figure 5.1: Only unknown features vehicle environment, first run

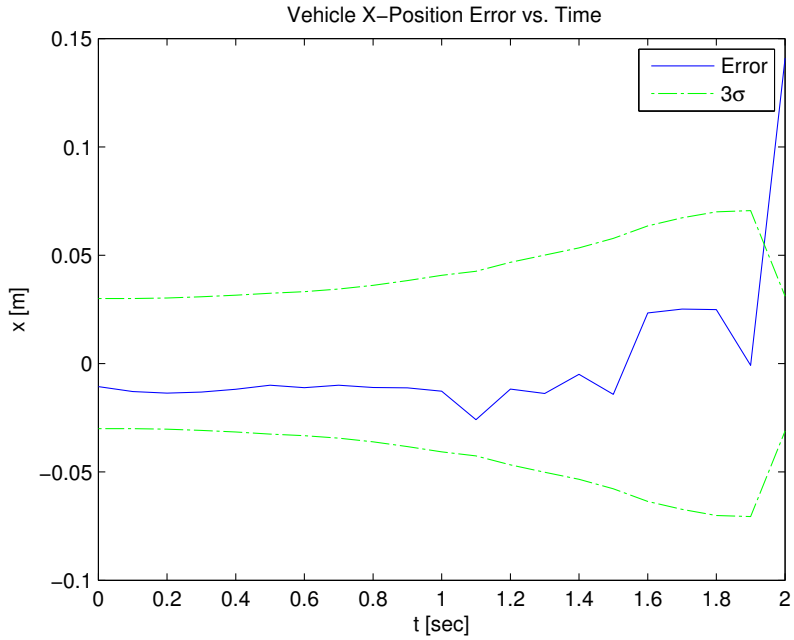


Figure 5.2: Only unknown features vehicle x-position error versus time, first run

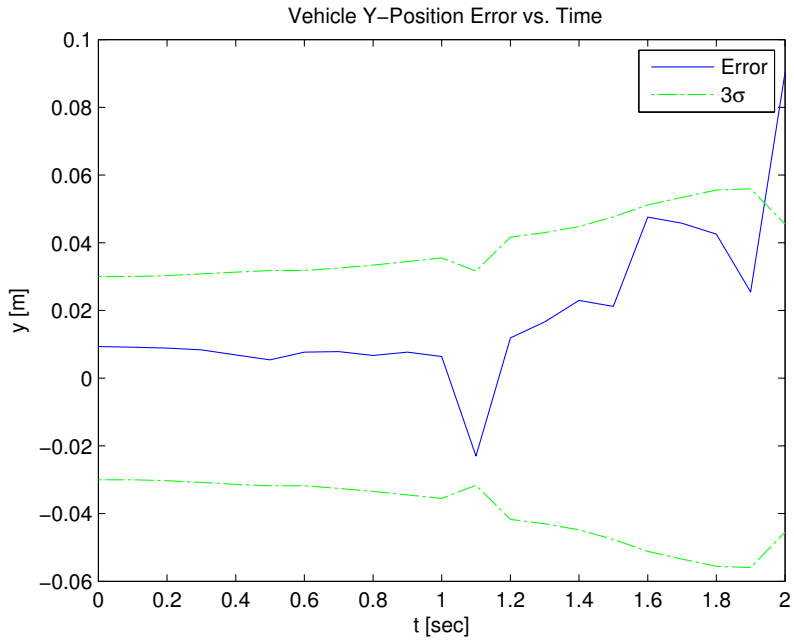


Figure 5.3: Only unknown features vehicle y-position error versus time, first run

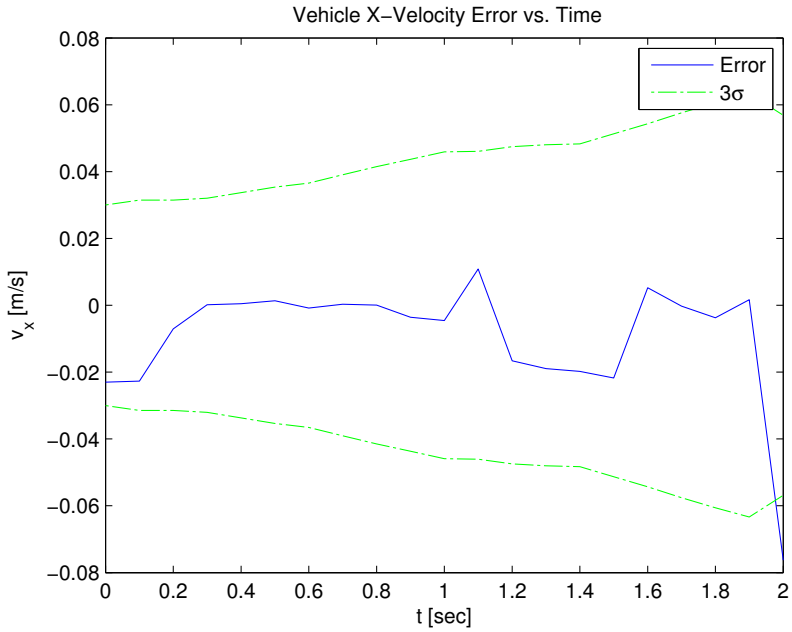


Figure 5.4: Only unknown features vehicle x-velocity error versus time, first run

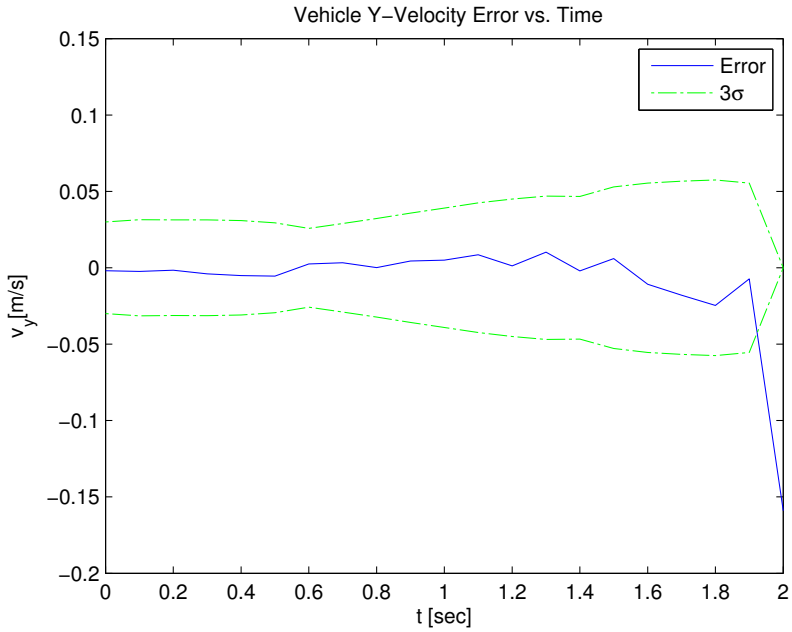


Figure 5.5: Only unknown features vehicle y-velocity error versus time, first run

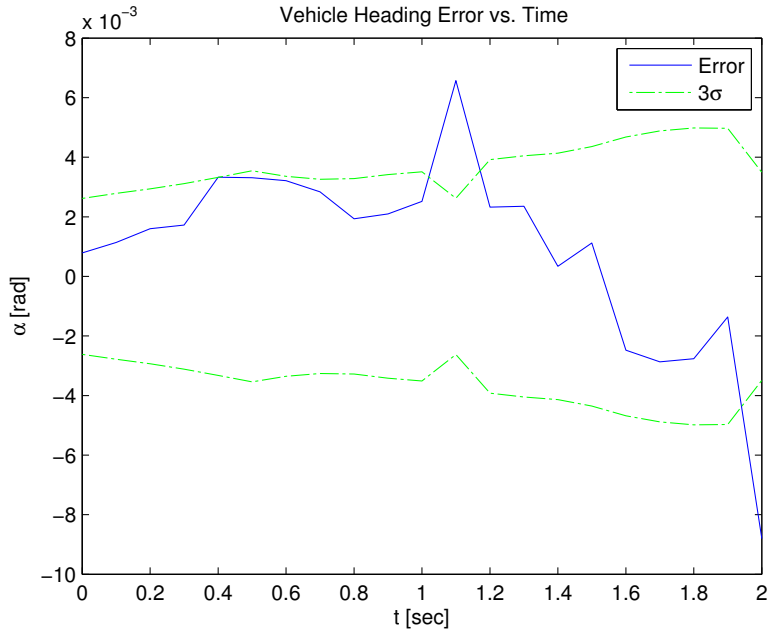


Figure 5.6: Only unknown features vehicle heading error versus time, first run

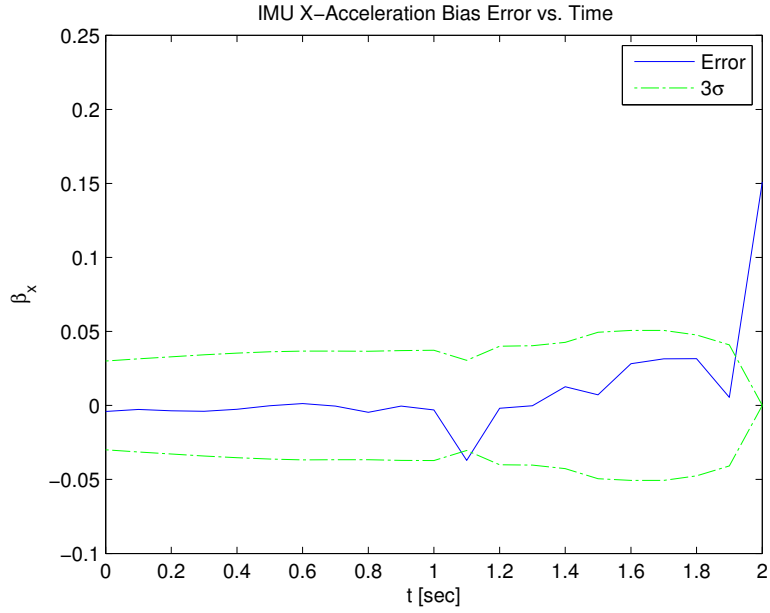


Figure 5.7: Only unknown features IMU x-accelerometer bias error versus time, first run

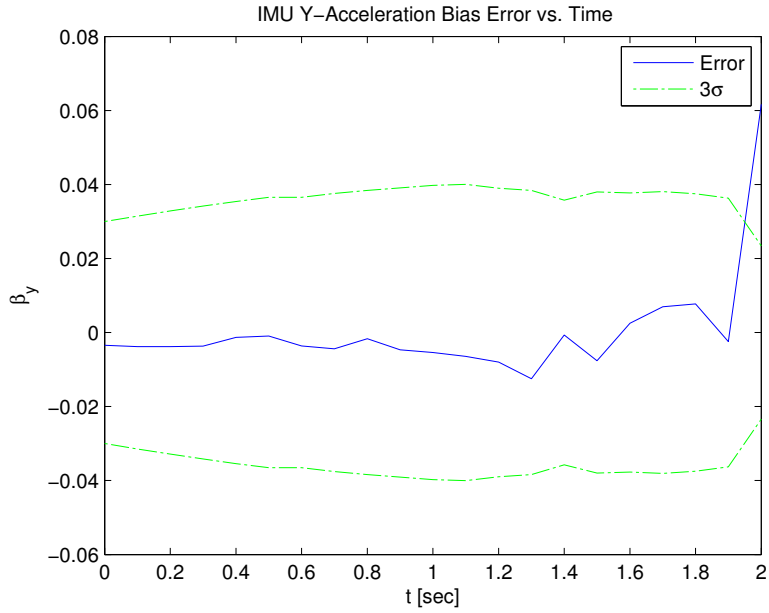


Figure 5.8: Only unknown features IMU y-accelerometer bias error versus time, first run

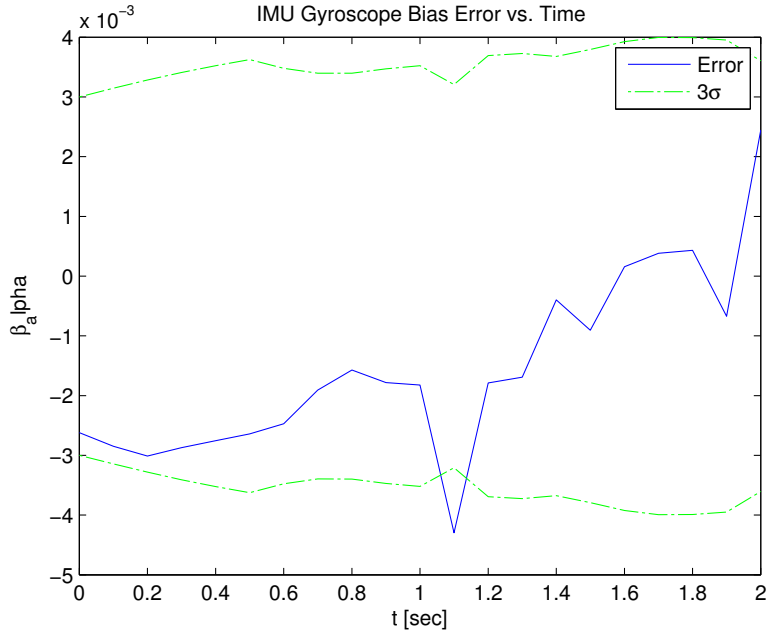


Figure 5.9: Only unknown features IMU gyroscopic bias error versus time, first run

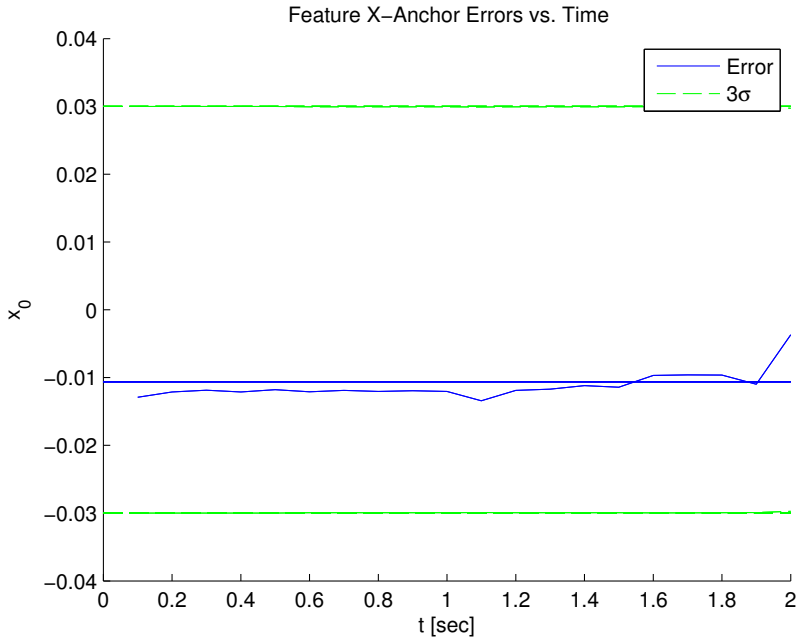


Figure 5.10: Only unknown features feature x-anchor errors versus time, first run

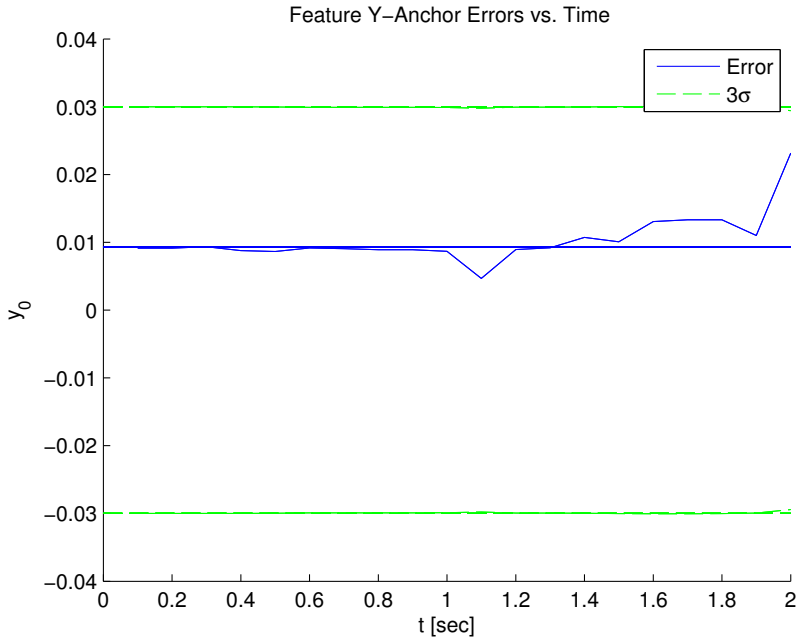


Figure 5.11: Only unknown features feature y-anchor errors versus time, first run

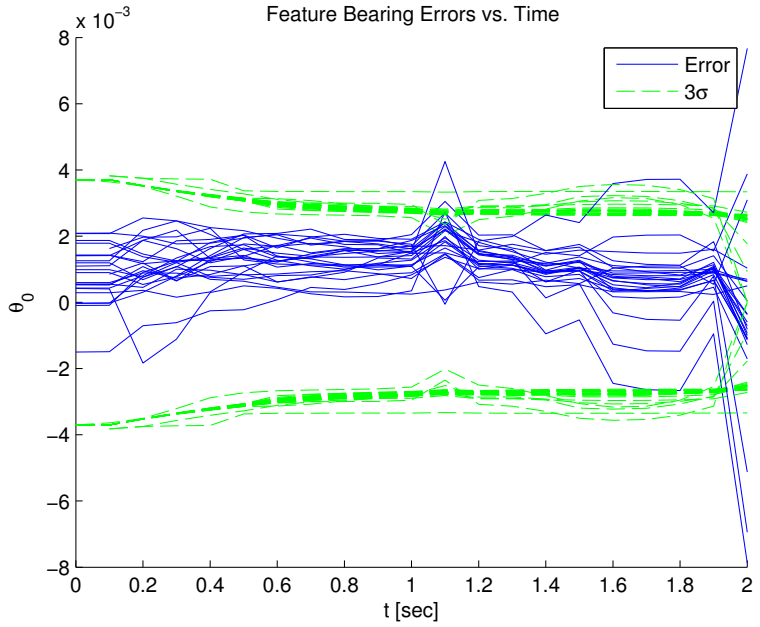


Figure 5.12: Only unknown features feature bearing errors versus time, first run

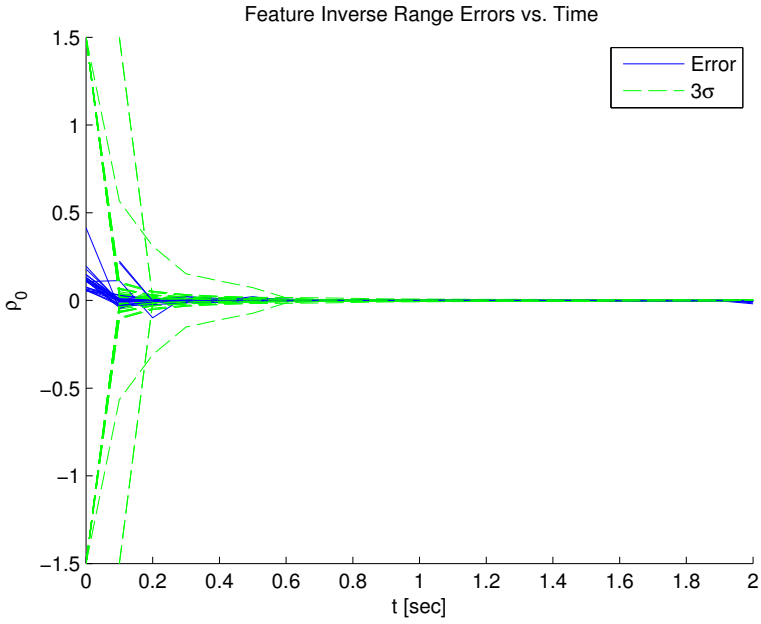


Figure 5.13: Only unknown features feature inverse-range errors versus time, first run

The first simulation run produces somewhat acceptable EKF estimation performance. In Figure 5.1 the vehicle estimated its own path seemingly well; the vehicle also estimated the unknown feature locations close to the truth. (The collection of open circles on the left-hand side of the plot are indicative of unknown features within the environment that were not seen by the vehicle camera, and thus not estimated.) However, there is visible error in the estimated locations of the unknown features. Most of the vehicle and feature state errors in Figures 5.2-5.13 remain within their respective 3σ bounds, though deviations do exist (indicative of an estimation performance lacking in accuracy).

The estimation accuracy exhibited by this simulation is not particularly desirable. Now examine the results of the second simulation run.

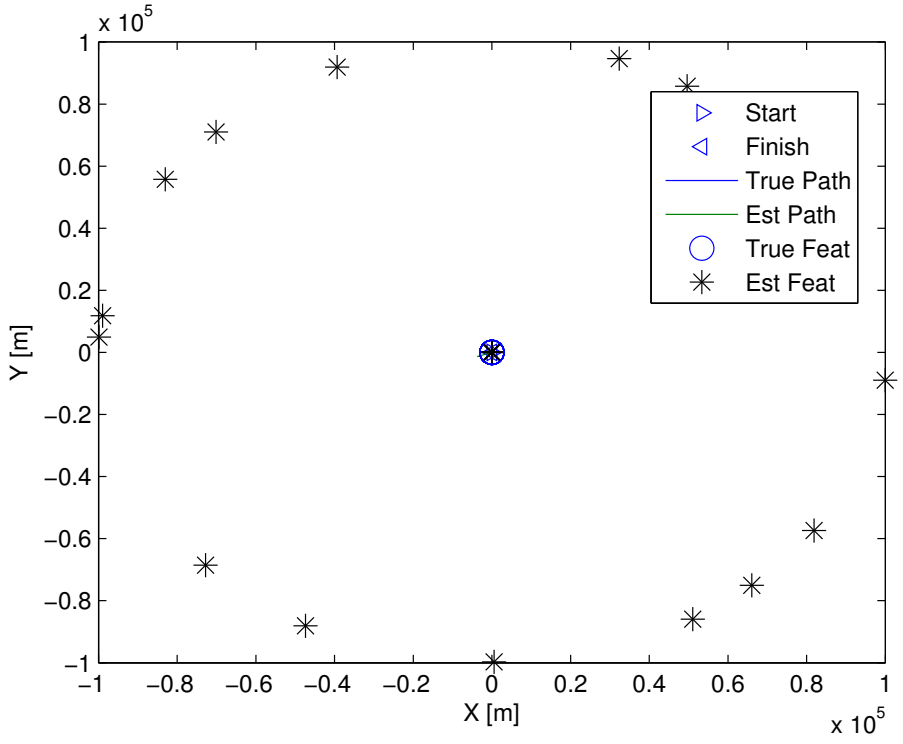


Figure 5.14: Only unknown features vehicle environment, second run

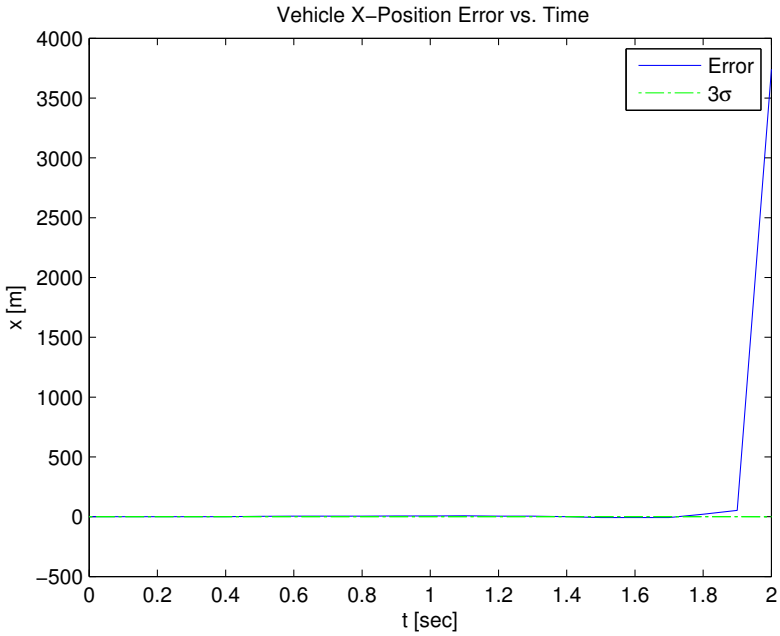


Figure 5.15: Only unknown features vehicle x-position error versus time, second run

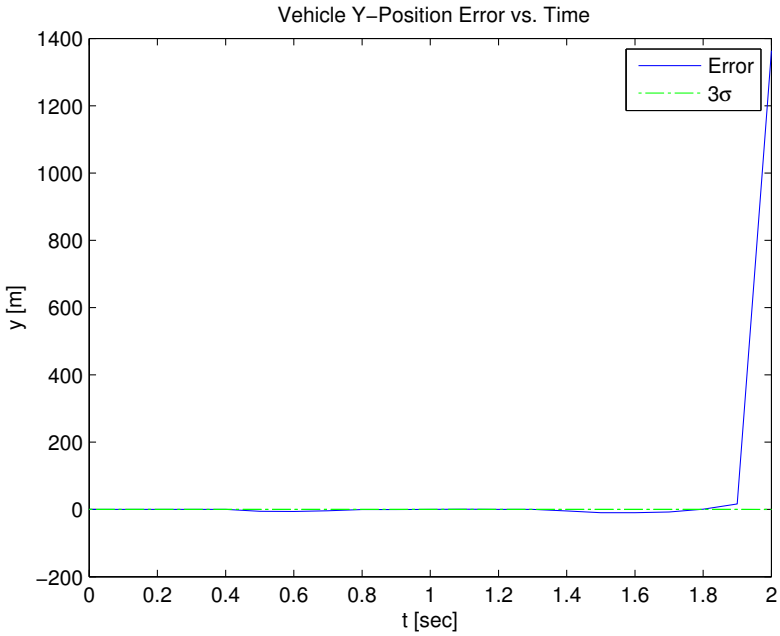


Figure 5.16: Only unknown features vehicle y-position error versus time, second run

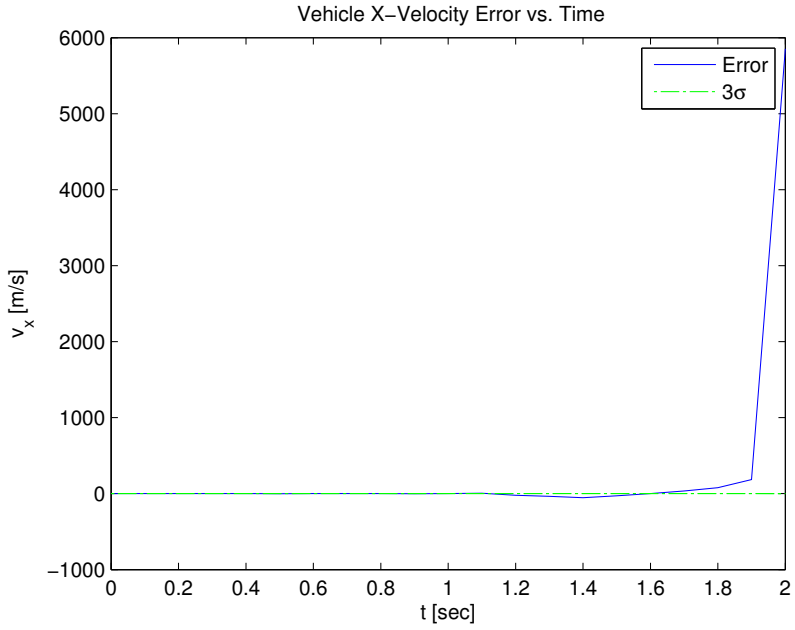


Figure 5.17: Only unknown features vehicle x-velocity error versus time, second run

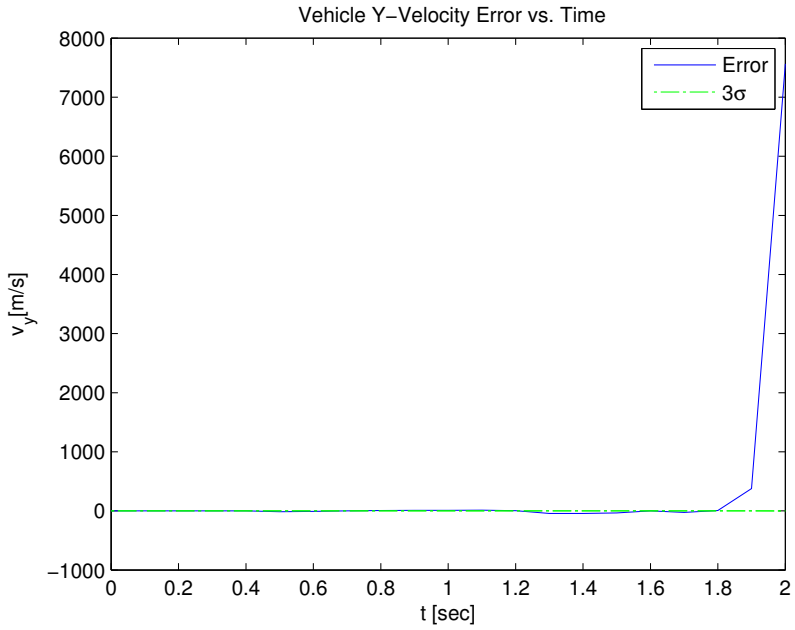


Figure 5.18: Only unknown features vehicle y-velocity error versus time, second run

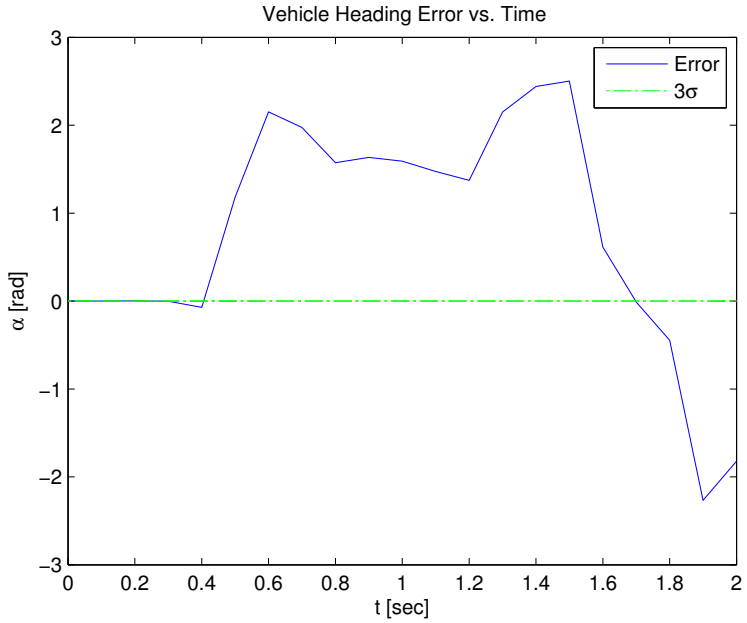


Figure 5.19: Only unknown features vehicle heading error versus time, second run

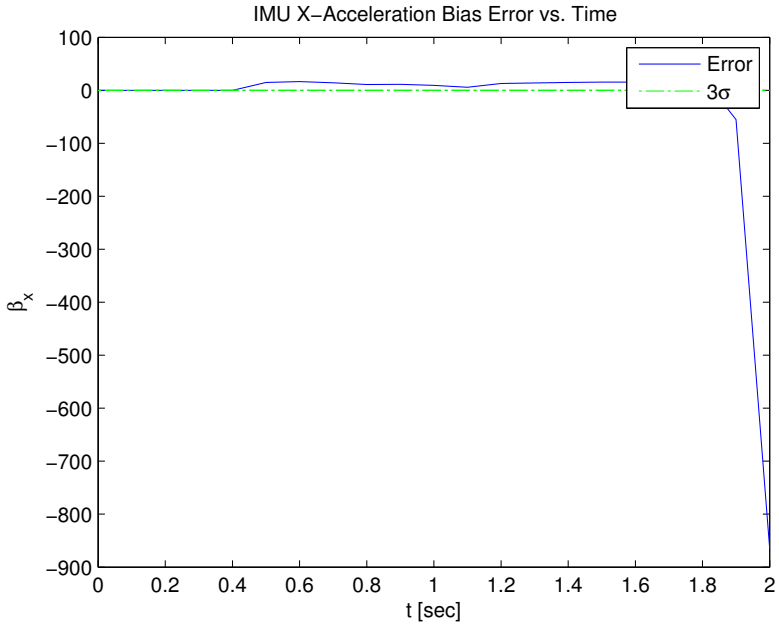


Figure 5.20: Only unknown features IMU x-accelerometer bias error versus time, second run

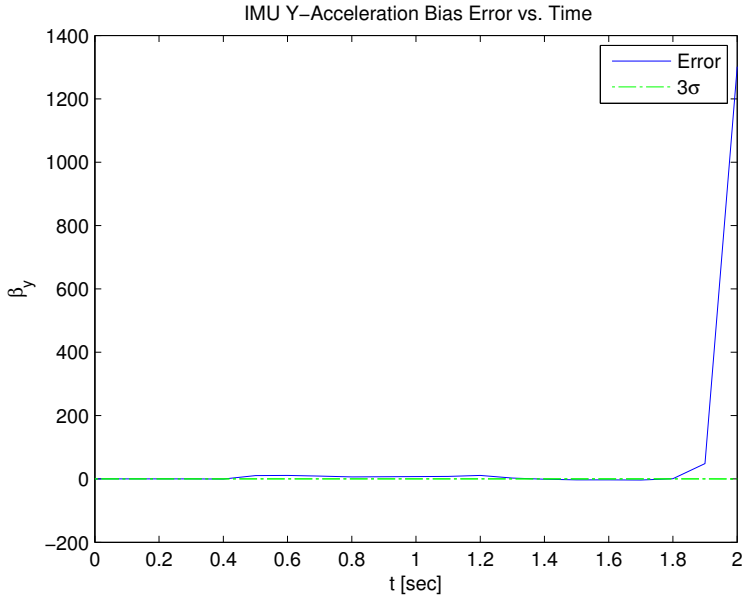


Figure 5.21: Only unknown features IMU y-accelerometer bias error versus time, second run

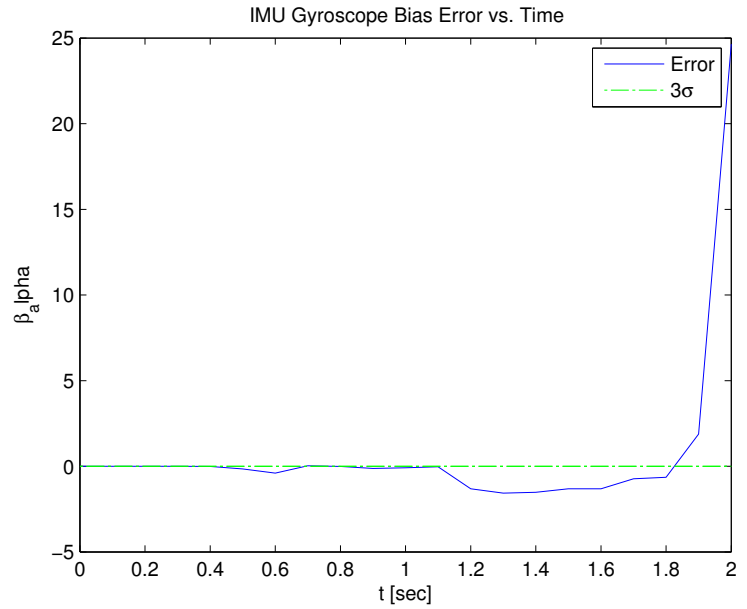


Figure 5.22: Only unknown features IMU gyroscopic bias error versus time, second run

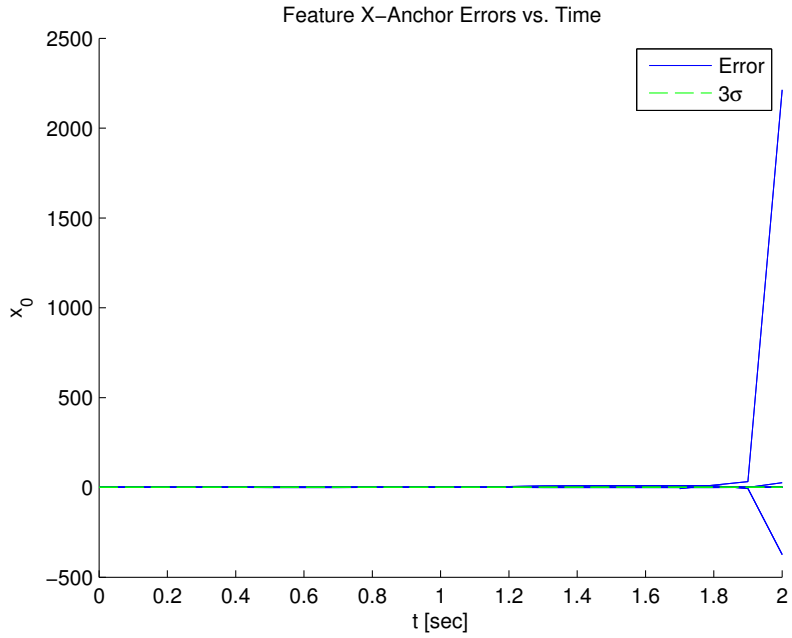


Figure 5.23: Only unknown features feature x-anchor errors versus time, second run

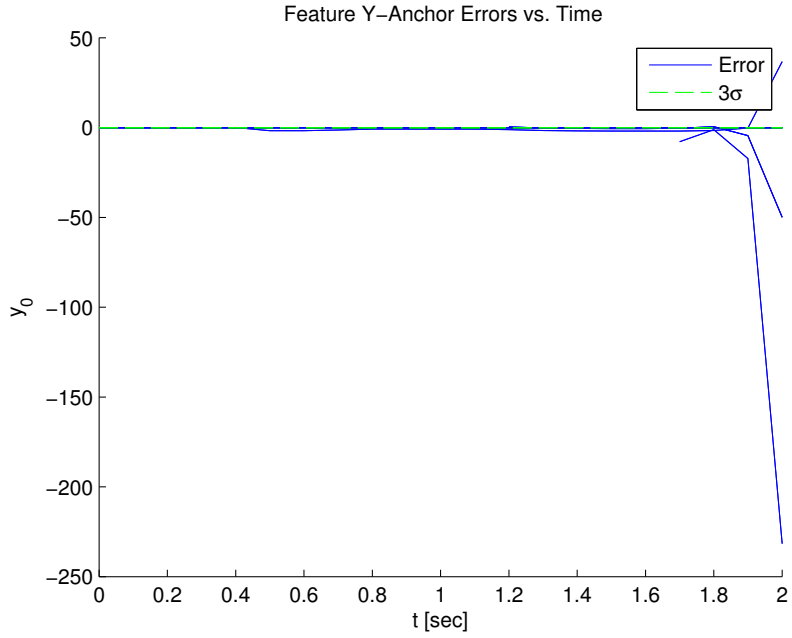


Figure 5.24: Only unknown features feature y-anchor errors versus time, second run

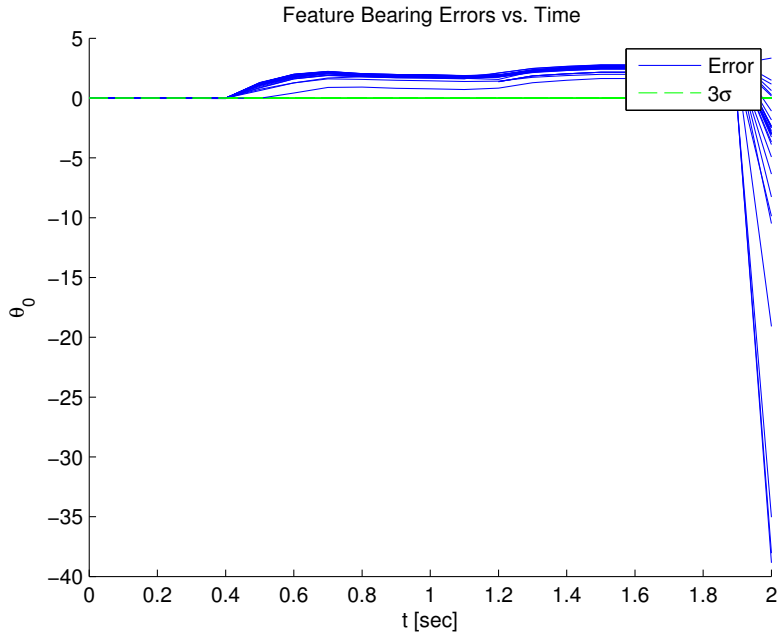


Figure 5.25: Only unknown features feature bearing errors versus time, second run

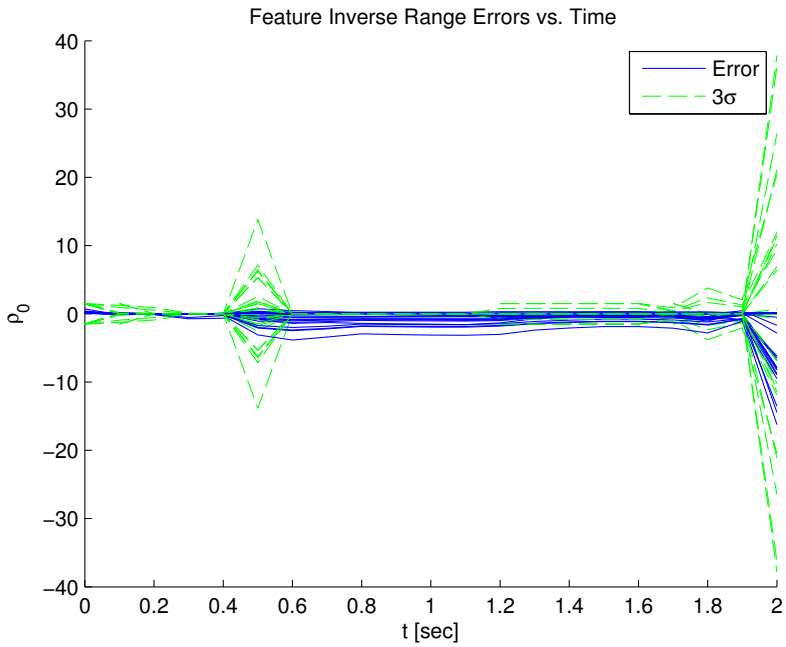


Figure 5.26: Only unknown features feature inverse-range errors, second run

The second simulation run produces extremely poor EKF estimation performance. In Figure 5.14, all of the unknown feature position estimates are very far from the truth, and the true and estimated vehicle paths diverge extremely far from each other (note the scale of the x and y-axes). Further, the vehicle and feature state errors shown in Figures 5.15-5.26 diverge significantly from their respective 3σ bounds. These results are indicative of EKF divergence.

The reason for the undesirable performance of both simulations is the lack of global information in the system. If no known features are present in the environment, then all of the information available to the vehicle is relative information comprised of estimations and measurements, both of which contain noise and uncertainty. This lack of global information is a pitfall which can cause the poor results from the second simulation. In the author's experience with simulations involving only unknown features, runs with acceptable estimation performance exist. However, most runs are similar to the two simulations shown in this section.

5.2 Environments with Known and Unknown Features

An environment containing only known features is too simplistic, giving very little useful insight; an environment containing only unknown features is too prone to poor estimation performance. However, an environment containing both known and unknown features provides interesting challenges while also providing global information and minimizing poor estimation performances. Additionally, it is realistic to assume an environment that a vehicle possesses some limited information about. Thus, the rest of this thesis is concerned with environments containing both known and unknown features.

An OPG MonoSLAM simulation with both known and unknown features is performed using the same system parameters from the previous simulations in this

section. Figures 5.27 and 5.28 display the vehicle environment with the resulting optimal path.

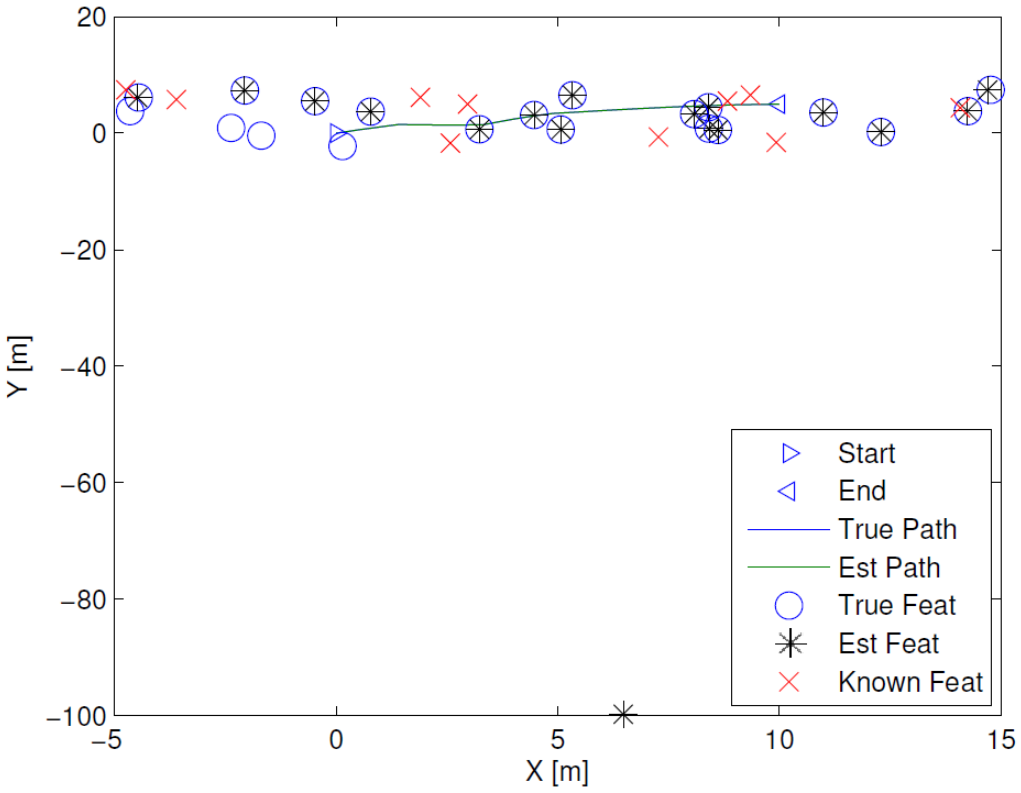


Figure 5.27: MonoSLAM with OPG vehicle environment

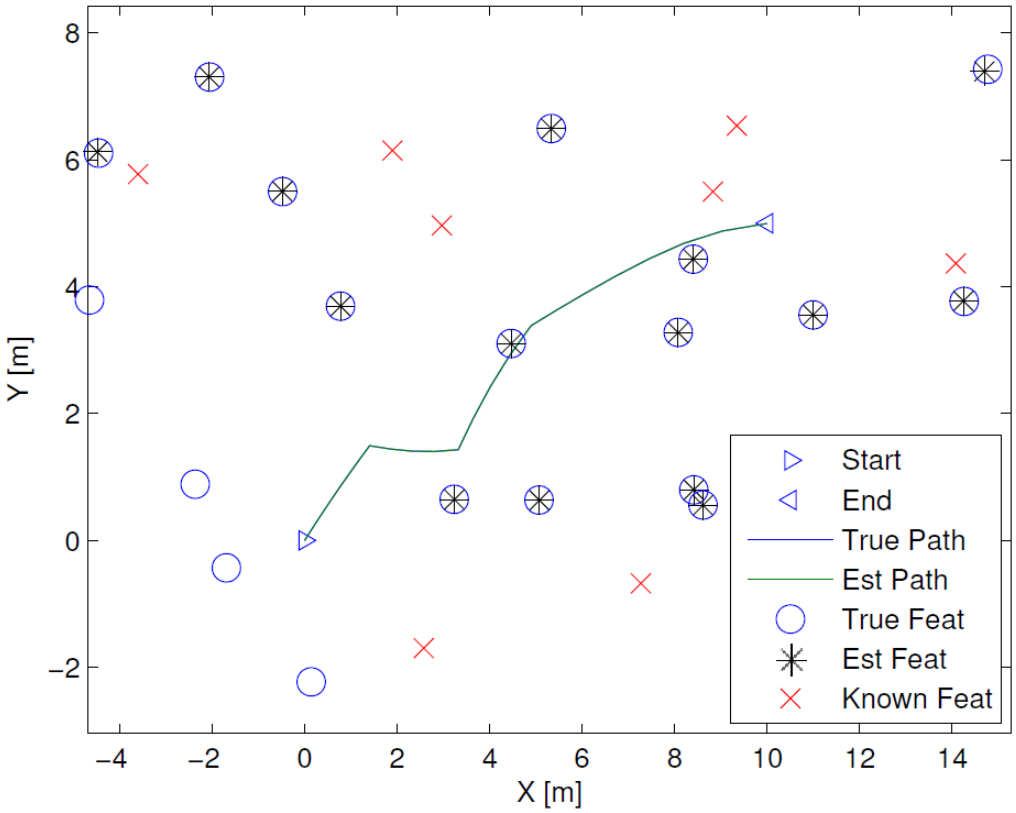


Figure 5.28: MonoSLAM with OPG vehicle environment (zoomed)

The environment shown in Figure 5.27 is zoomed out because of one outlying unknown feature position estimate. This outlier is a direct result of the corresponding true feature being seen only once. This is not a failure of the MonoSLAM algorithm; instead it is simply a consequence of limited camera visibility. Figure 5.28 gives a closer look at the densely populated area in Figure 5.27. The four open circles indicate four unknown features whose positions were not estimated correctly: the one feature previously discussed and three features which were never seen by the vehicle. The presence of known features in the environment helps EKF estimation performance, as is evidenced by the well-localized unknown feature location estimates

and well-estimated path.

In order to examine estimation performance more closely, view the vehicle state error plots in Figures 5.29-5.36 and the feature state errors shown in Figures 5.37-5.40.

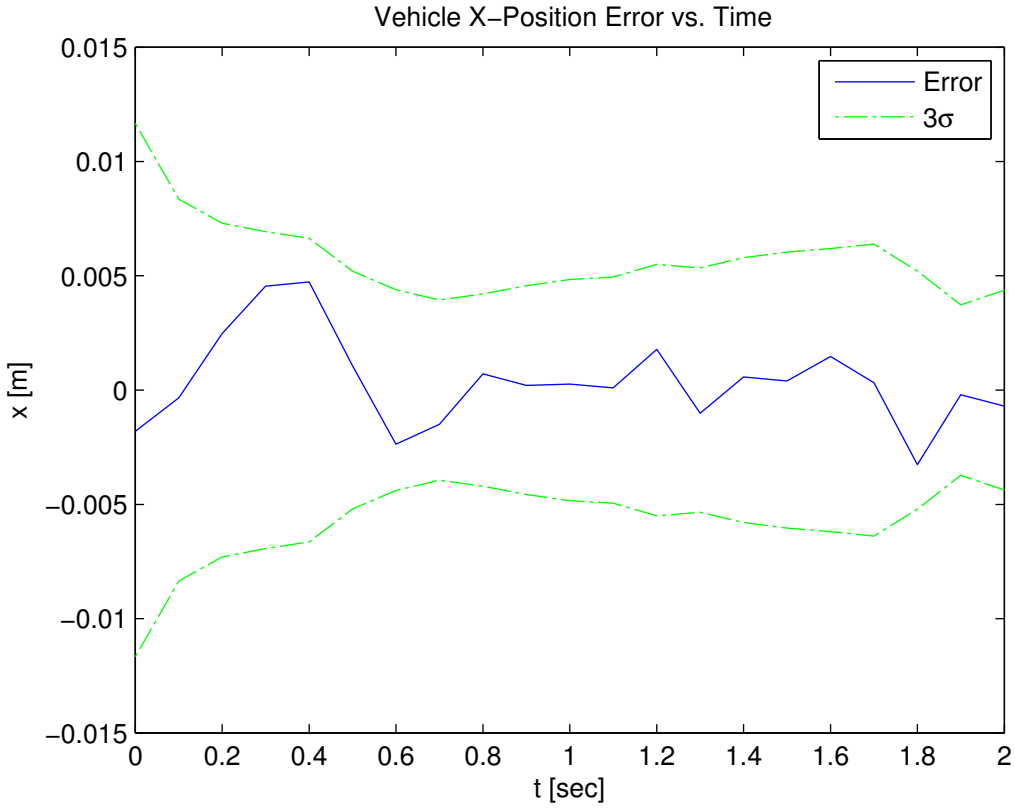


Figure 5.29: MonoSLAM with OPG vehicle x-position error versus time

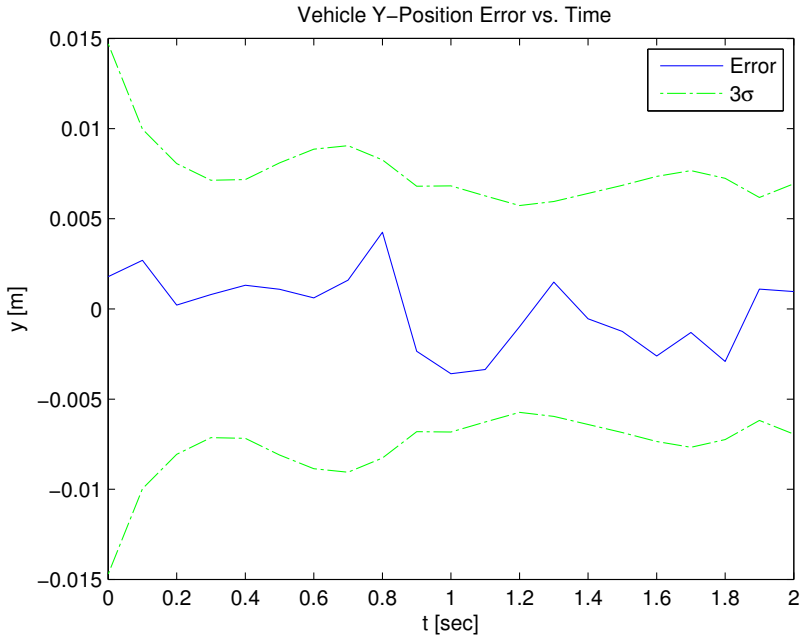


Figure 5.30: MonoSLAM with OPG vehicle y-position error versus time

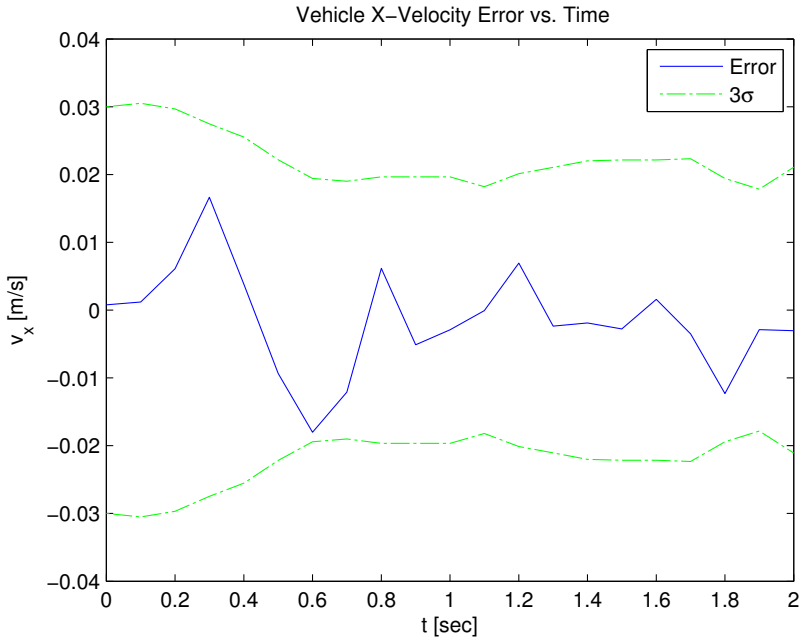


Figure 5.31: MonoSLAM with OPG vehicle x-velocity error versus time

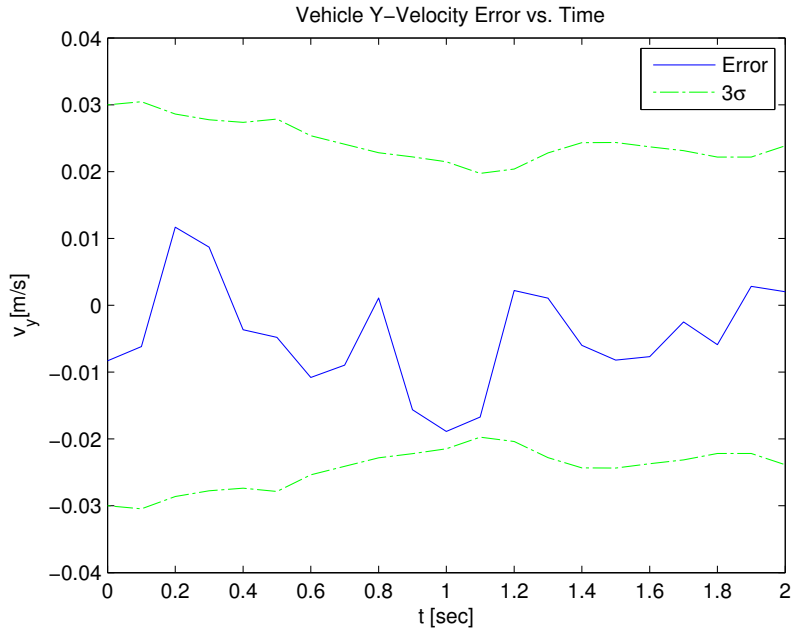


Figure 5.32: MonoSLAM with OPG vehicle y-velocity error versus time

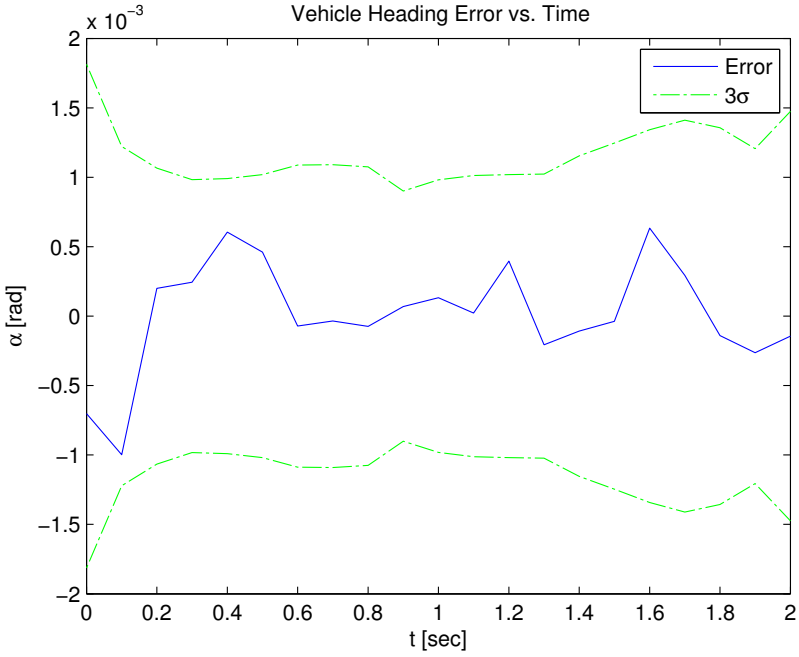


Figure 5.33: MonoSLAM with OPG vehicle heading error versus time

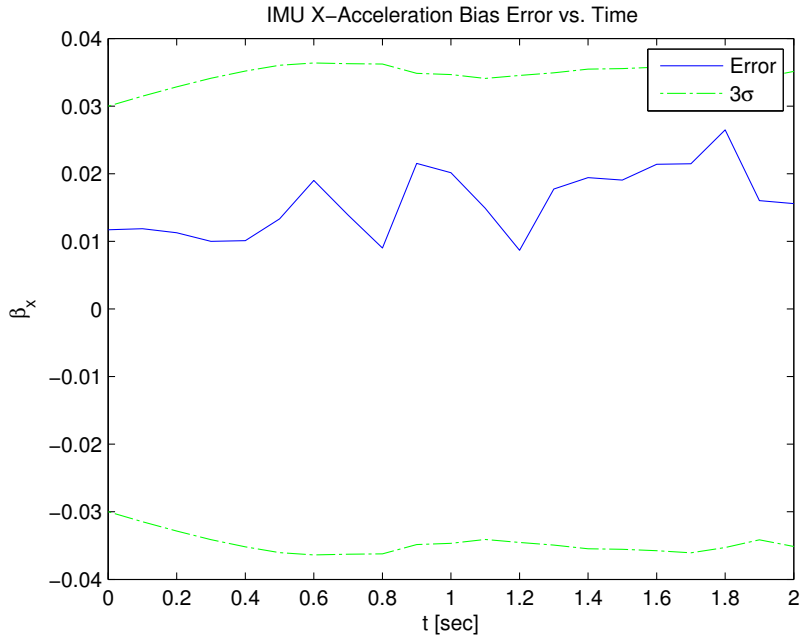


Figure 5.34: MonoSLAM with OPG IMU x-accelerometer bias error versus time

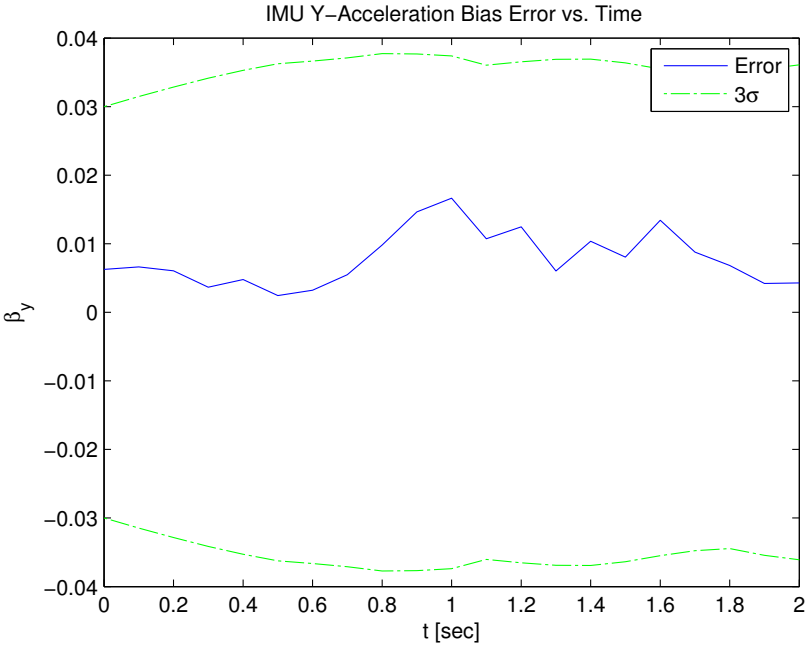


Figure 5.35: MonoSLAM with OPG IMU y-accelerometer bias error versus time

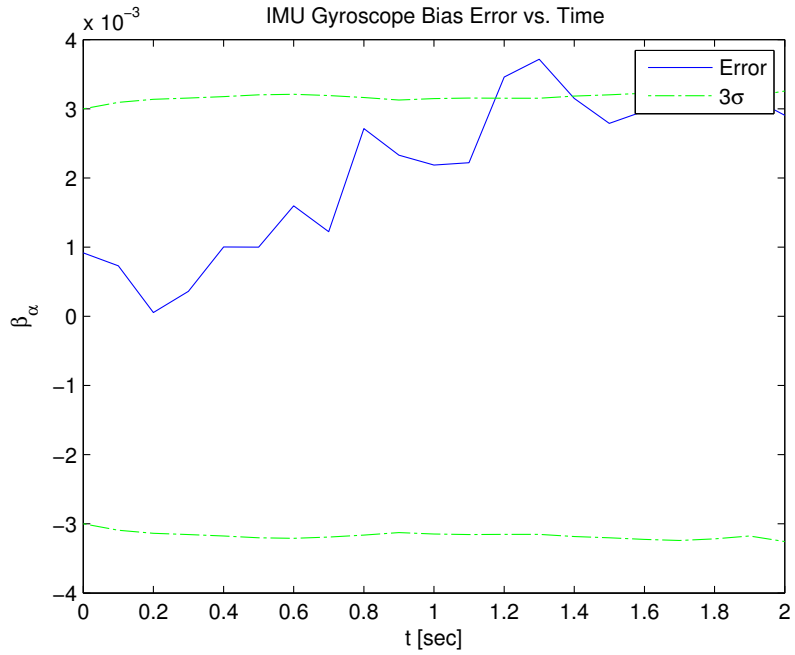


Figure 5.36: MonoSLAM with OPG IMU gyroscopic bias error versus time

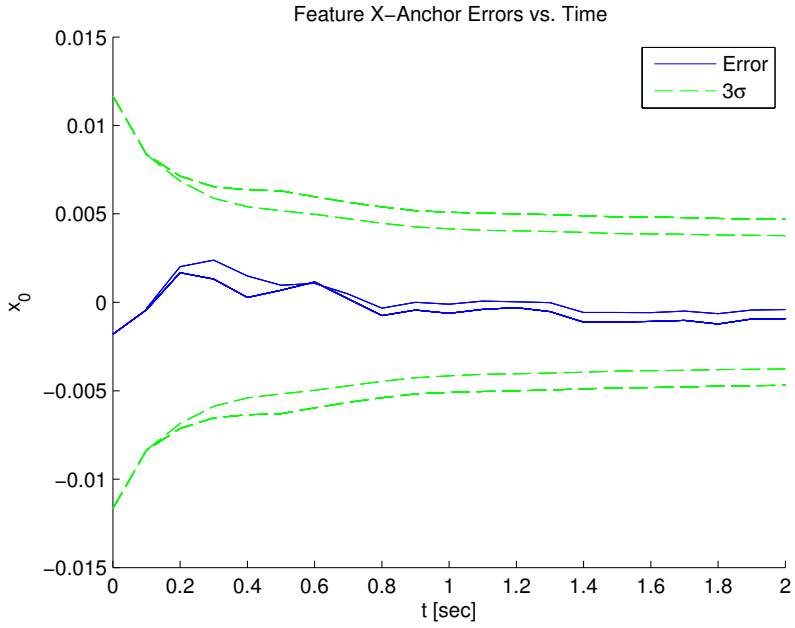


Figure 5.37: MonoSLAM with OPG feature x-anchor errors versus time

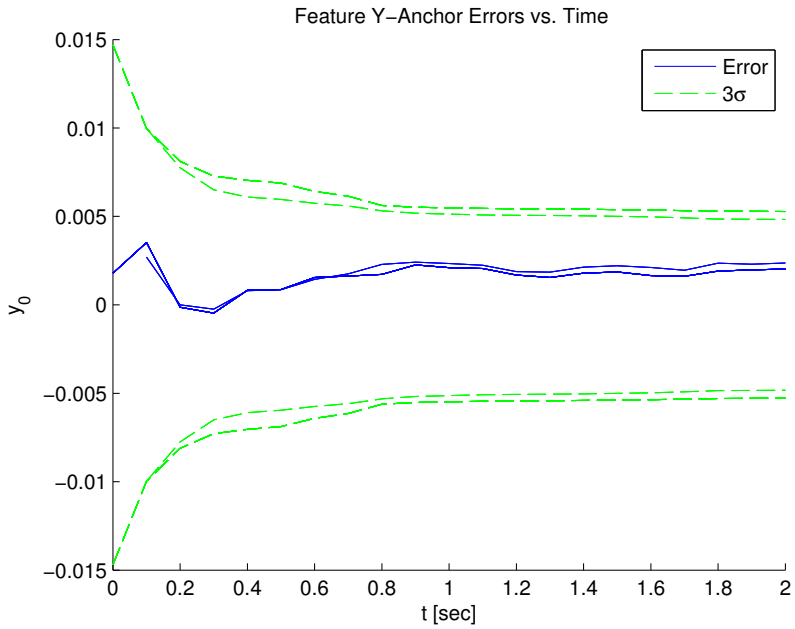


Figure 5.38: MonoSLAM with OPG feature y-anchor errors versus time

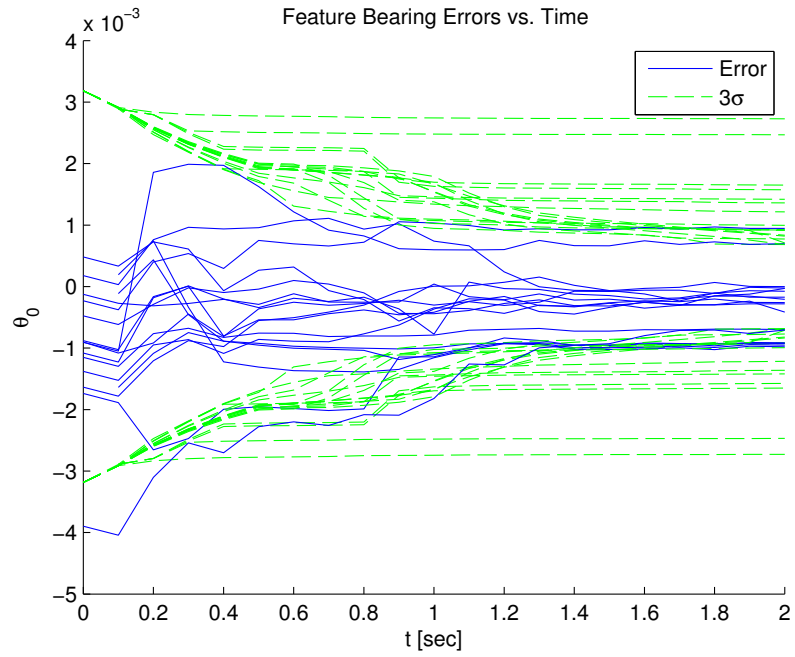


Figure 5.39: MonoSLAM with OPG feature bearing errors versus time

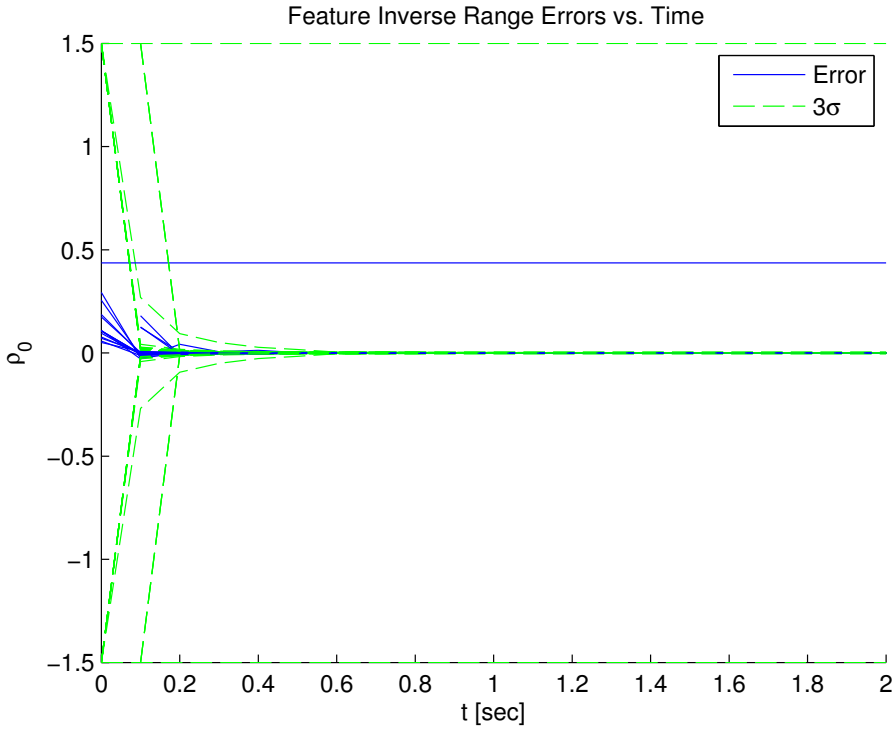


Figure 5.40: MonoSLAM with OPG feature inverse-range errors versus time

Most of the vehicle and features state errors are well within their respective 3σ bounds, though some 3σ violations do exist for the IMU gyroscopic bias error in Figure 5.36 and also for one unknown feature's bearing error in Figure 5.39. The fact that only a very small amount of data points violate the 3σ uncertainty boundaries indicates that while this simulation does not provide the highest level possible of estimation accuracy, it does provide an acceptable accuracy.

Observe the singular large feature inverse range error line in Figure 5.40. The fact that this error (and its 3σ bounds) do not change during the course of the simulation is indicative of no feature visibility after initialization. That is, the vehicle camera managed to see the feature once and initialize it but failed to make any successive bearing measurements. This is the same feature as the outlier in Figure 5.27.

The performance of the OPG algorithm on an environment comprised of both known and unknown features is vastly superior to OPG performance on environments with only unknown features. This confirms the author's decision to focus on environments with both known and unknown features.

6. COMPARISON OF OPTIMAL PATH GENERATION WITH TYPICAL MONOSLAM PATH

A typical MonoSLAM user-selected vehicle path is the sinusoid[7]. In this section, the sinusoid path will be used as a baseline for determining the merits of the optimal path generation scheme that has been developed in this thesis. A simulation was performed in order to compare the performance of MonoSLAM with the typical sinusoidal path with the performance of the OPG algorithm. The vehicle start and finish positions, all feature locations, and all initial state estimates are identical between the two methods in the simulation.The simulation runs for 2.2 seconds. The sinusoidal path uses a timestep of 0.01 seconds; the OPG path uses a timestep of 0.1 seconds. (This difference in time step is to ensure that the sinusoidal curve is sampled often enough so that the resulting path is smooth, without jagged segments.) The vehicle camera has a field of view of 180° that extends out to 100 meters. The unknown feature change in estimated position parameter, σ_f, is 10 meters, and unknown features must be seen 4 times previously in order to be eligible for cost function inclusion. The cost function is reformulated every 4 steps.

Figure 6.1 shows a MonoSLAM vehicle environment with a sinusoidal vehicle path. Figure 6.2 shows an optimal vehicle path with the same start/finish positions and feature locations as Figure 6.1. Note that in both simulations, the vehicle appears to have done a good job of estimating its path and the unknown feature locations. However, qualitatively, the optimal path is a much simpler path for the vehicle to travel on.

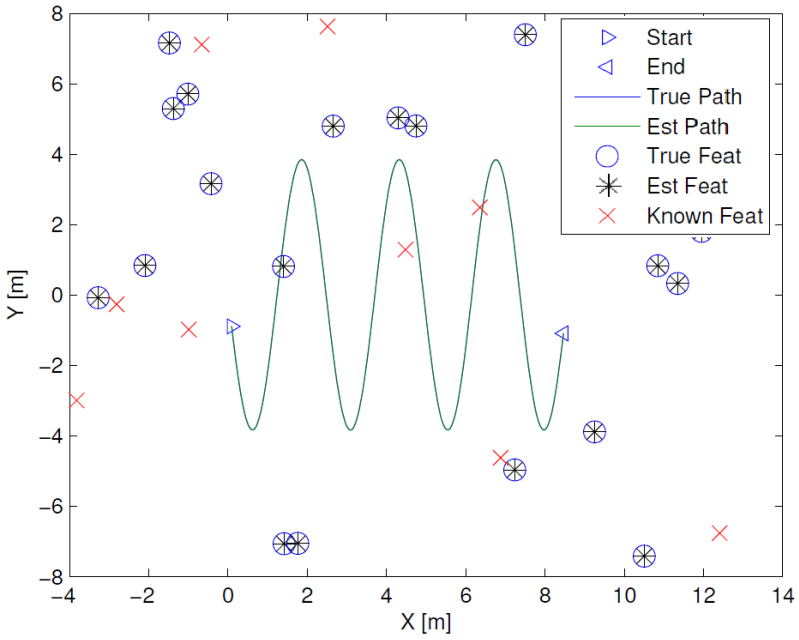


Figure 6.1: Sinusoidal MonoSLAM path

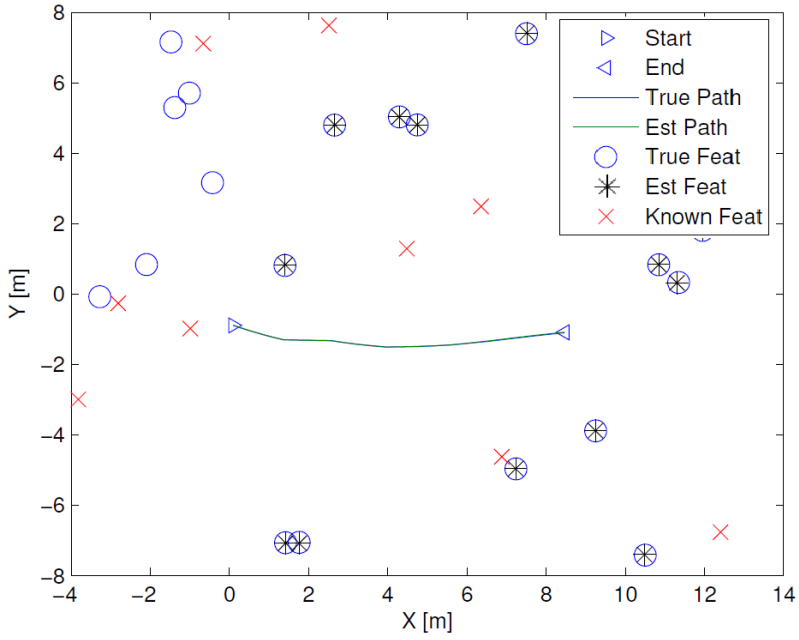


Figure 6.2: Optimal MonoSLAM path

6.1 Fuel Usage

In order to qualitatively examine the fuel usage by each path, examine the acceleration magnitudes plot show in Figure 6.3. Clearly, the optimal path generation scheme produces vehicle accelerations far below those of the typical sinusoidal path. To quantify this acceleration disparity, the author integrates the acceleration magnitude plots according to the following integral (fuel integral):

$$\frac{1}{2}\int_{t_0}^{t_f} \|\mathbf{a}\|\,\mathrm{d}t$$

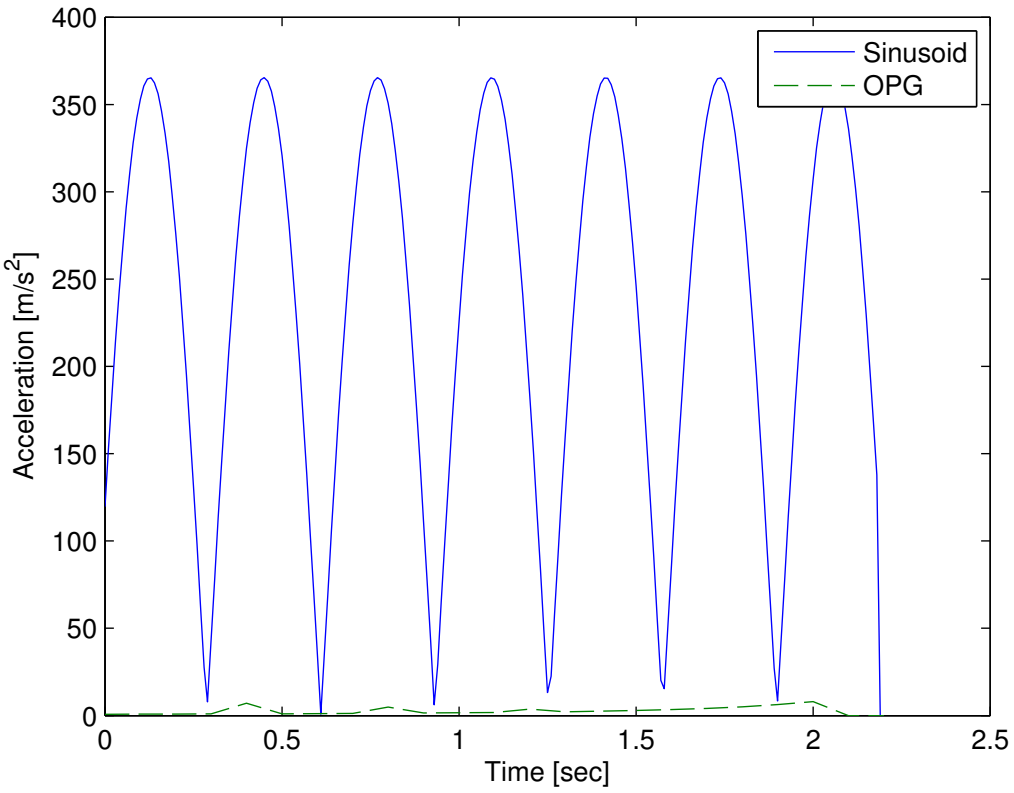


Figure 6.3: Vehicle acceleration magnitudes

Table 6.1: Fuel integral data

	OPG	Sinusoid	Units	Ratio
Fuel Integral	3.0648	260.0862	$\frac{\mathrm{m}}{s}$	0.0118

Table 6.1 gives the results of numerically integrating the fuel integral for both simulations. The fuel cost of the OPG simulation is a mere 1.18% of the fuel cost in the typical sinusoidal path simulation. This represents a staggering fuel savings.

6.2 Estimation Accuracy

Now examine the state estimation accuracy between the two simulations. Position errors are shown in Figures 6.4-6.7; velocity errors are shown in Figures 6.8-6.11; heading errors are shown in Figure 6.12-6.13; IMU bias errors are shown in Figures 6.15-6.19; feature anchor point errors are shown in Figures 6.20-6.23; feature bearing errors are shown in Figures 6.24 and 6.25; feature inverse-range errors are shown in 6.26 and 6.27. In all of the proceeding figures, the sinusoidal simulation results are located on the top half of the page in black and red, and the OPG results are located on the bottom half of the page in blue and green.

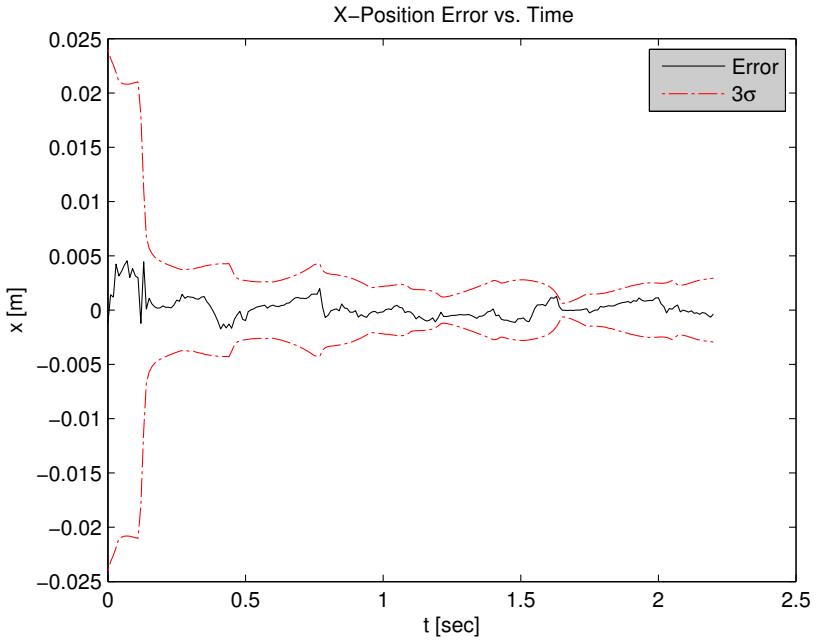


Figure 6.4: Sinusoidal MonoSLAM vehicle x-position error versus time

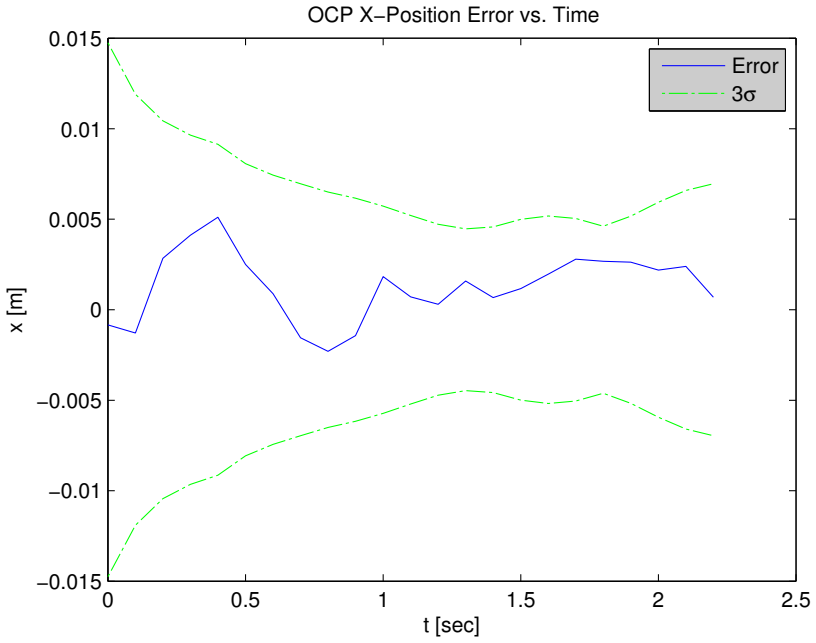


Figure 6.5: Optimal MonoSLAM vehicle x-position error versus time

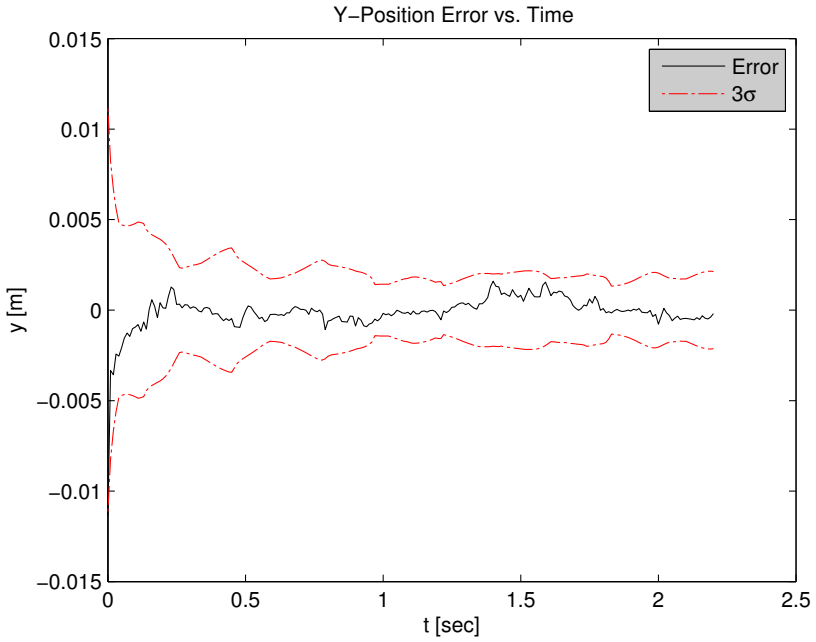


Figure 6.6: Sinusoidal MonoSLAM vehicle y-position error versus time

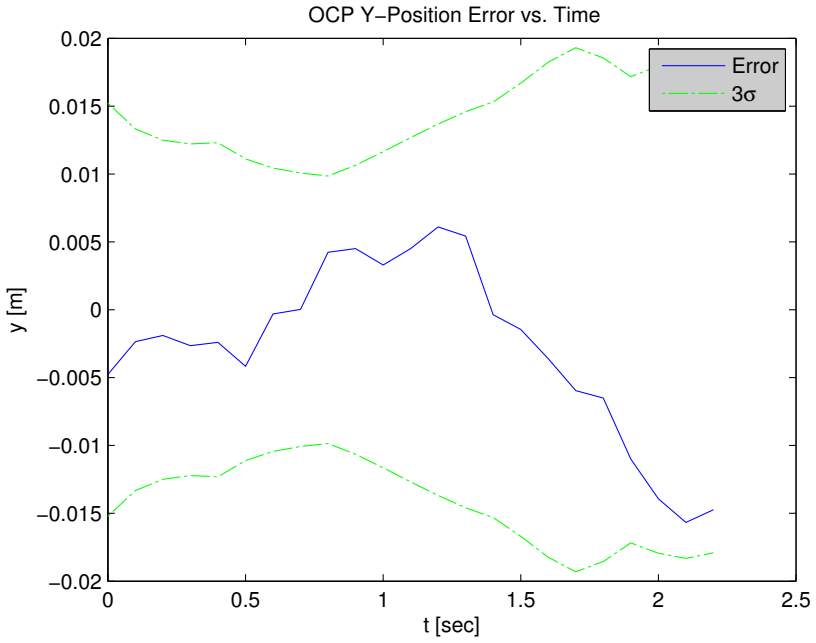


Figure 6.7: Optimal MonoSLAM vehicle y-position error versus time

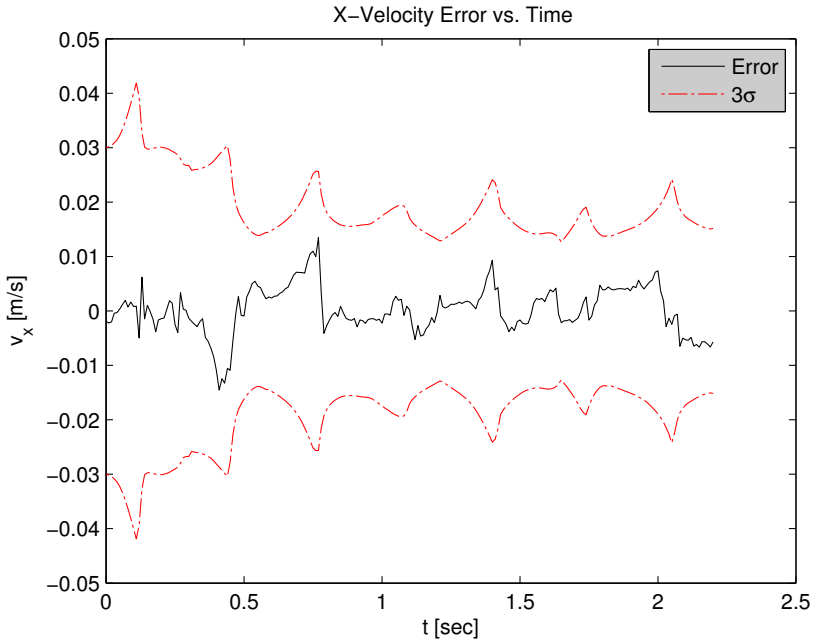


Figure 6.8: Sinusoidal MonoSLAM vehicle x-velocity error versus time

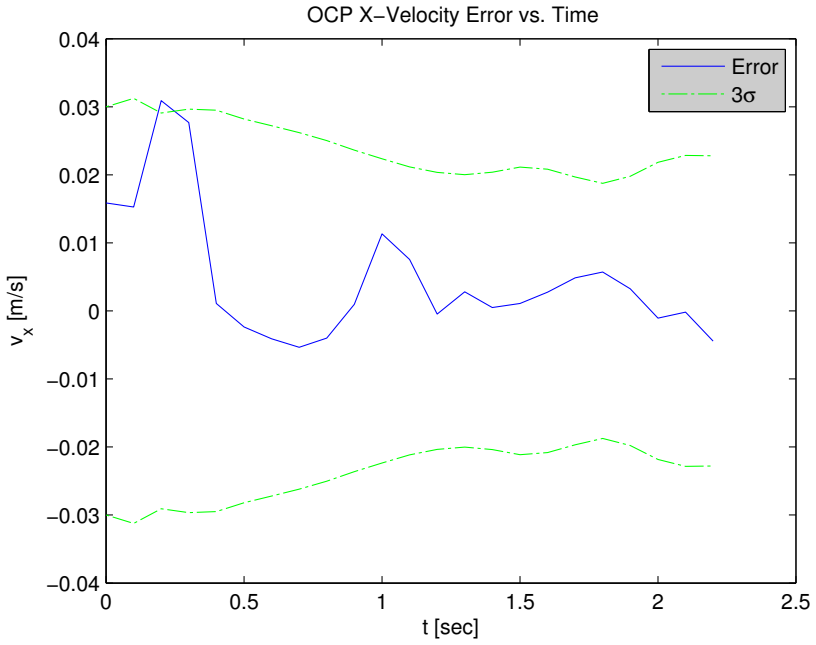


Figure 6.9: Optimal MonoSLAM vehicle x-velocity error versus time

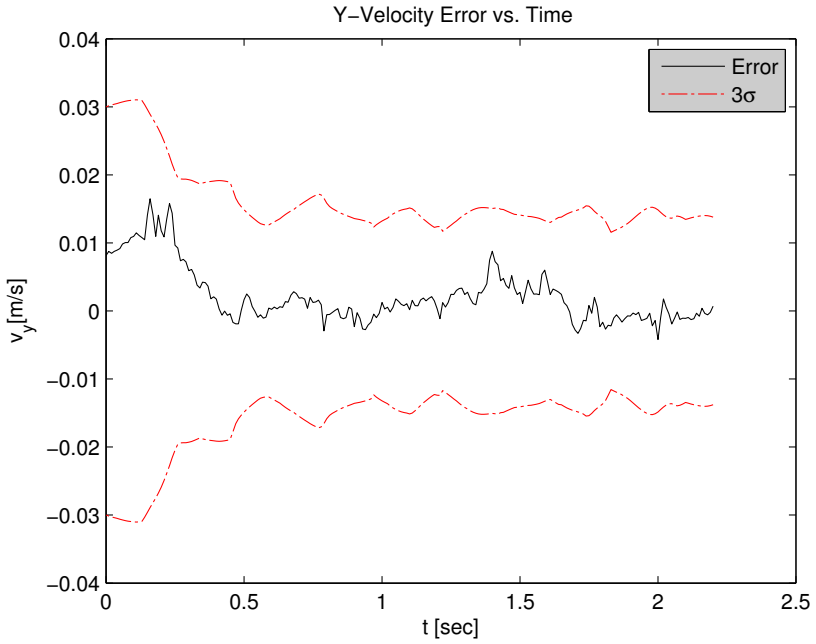


Figure 6.10: Sinusoidal MonoSLAM vehicle y-velocity error versus time

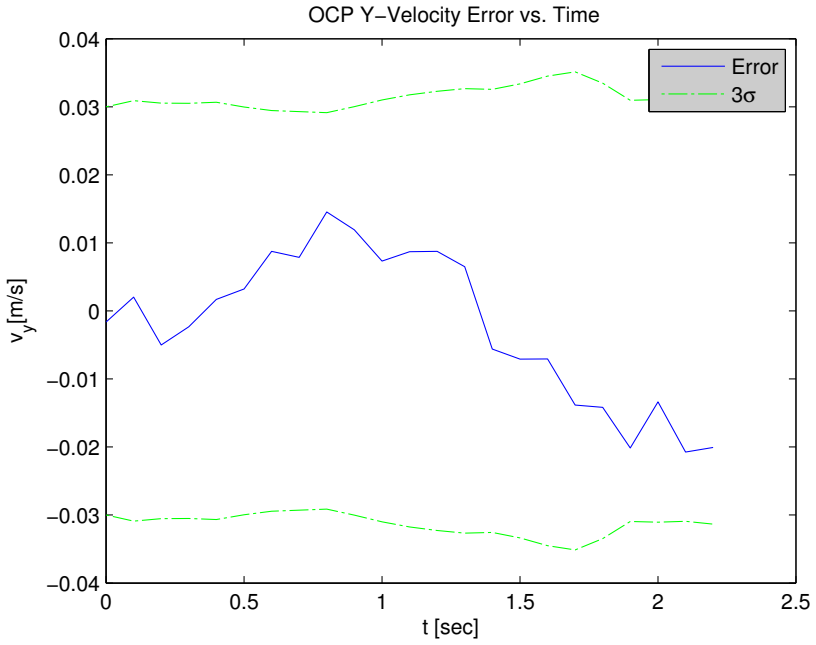


Figure 6.11: Optimal MonoSLAM vehicle y-velocity error versus time

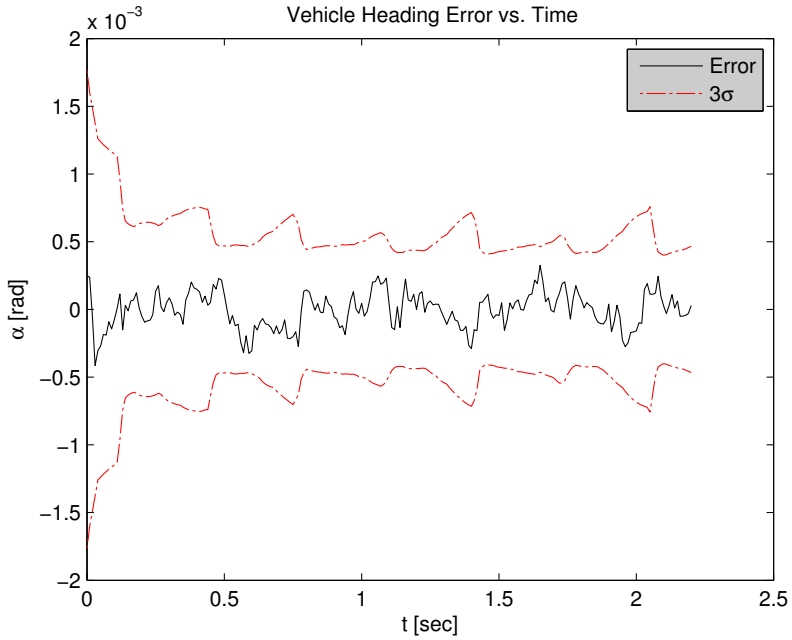


Figure 6.12: Sinusoidal MonoSLAM vehicle heading error versus time

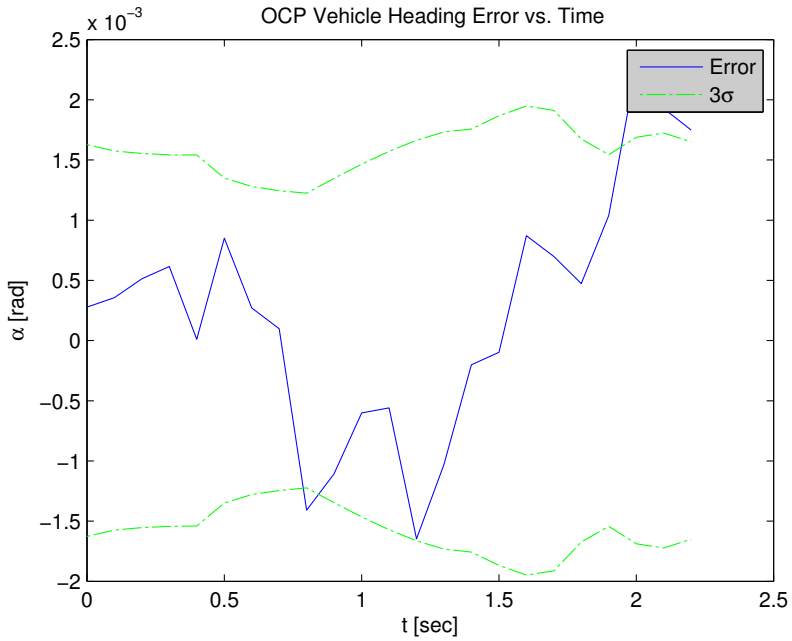


Figure 6.13: Optimal MonoSLAM vehicle heading error versus time

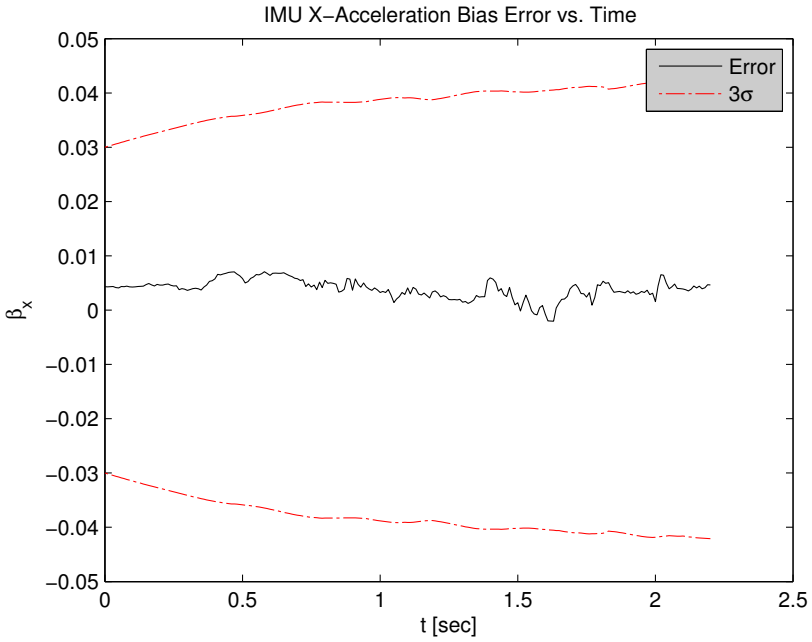


Figure 6.14: Sinusoidal MonoSLAM IMU x-accelerometer bias error versus time

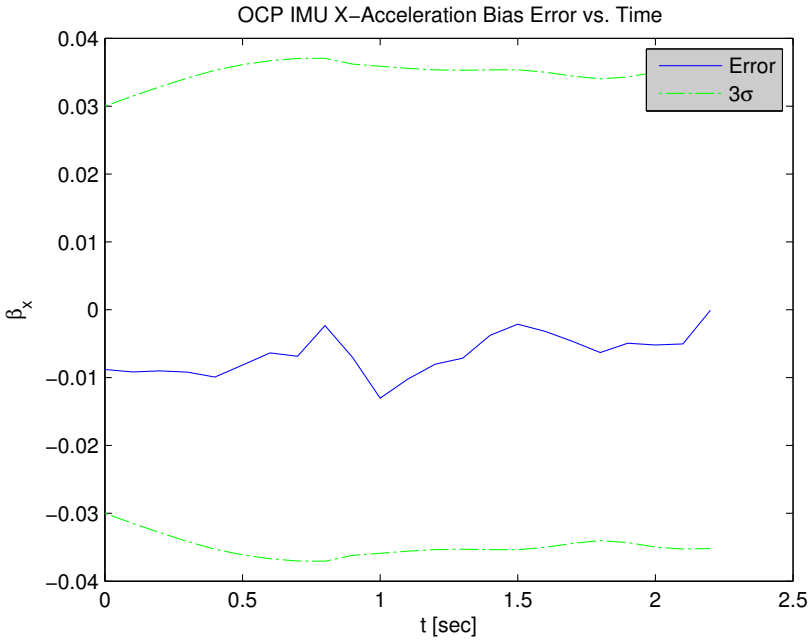


Figure 6.15: Optimal MonoSLAM IMU x-accelerometer bias error versus time

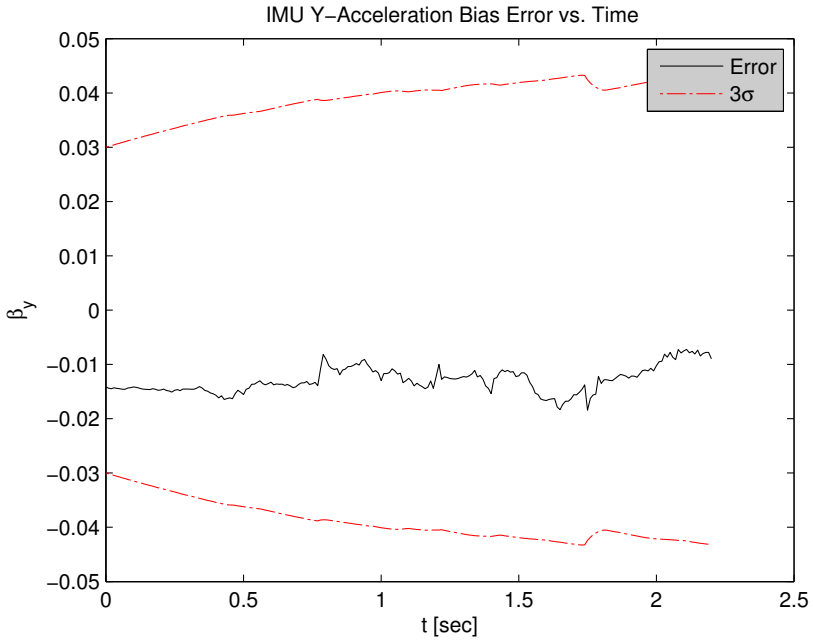


Figure 6.16: Sinusoidal MonoSLAM IMU y-accelerometer bias error versus time

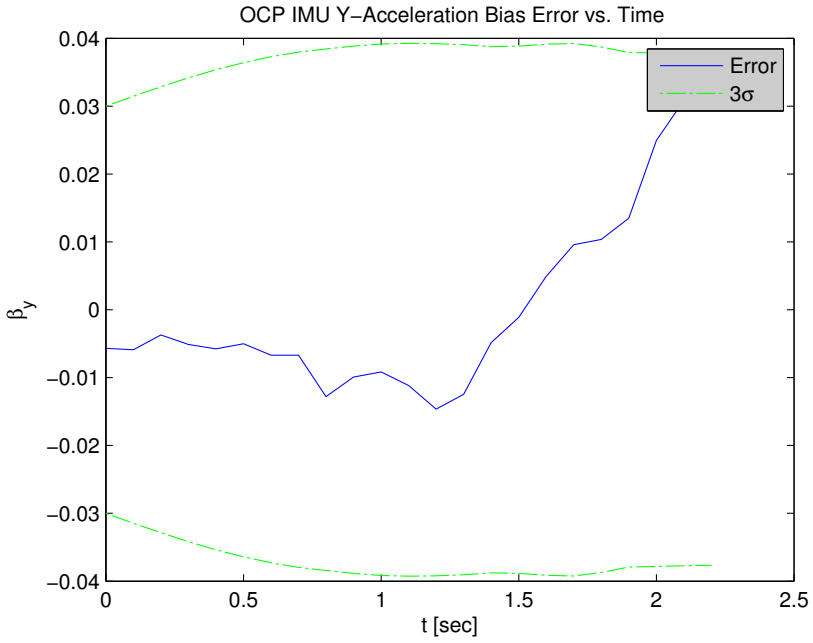


Figure 6.17: Optimal MonoSLAM IMU y-accelerometer bias error versus time

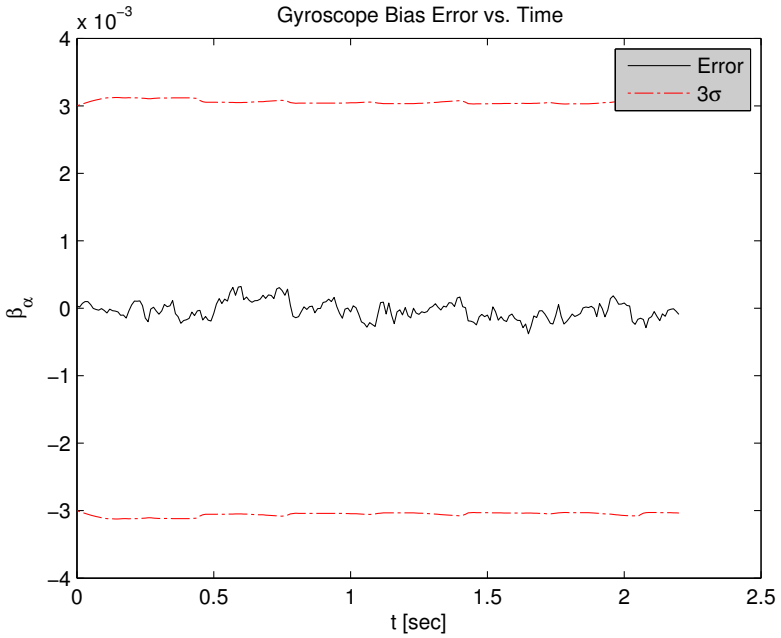


Figure 6.18: Sinusoidal MonoSLAM IMU gyroscopic bias error versus time

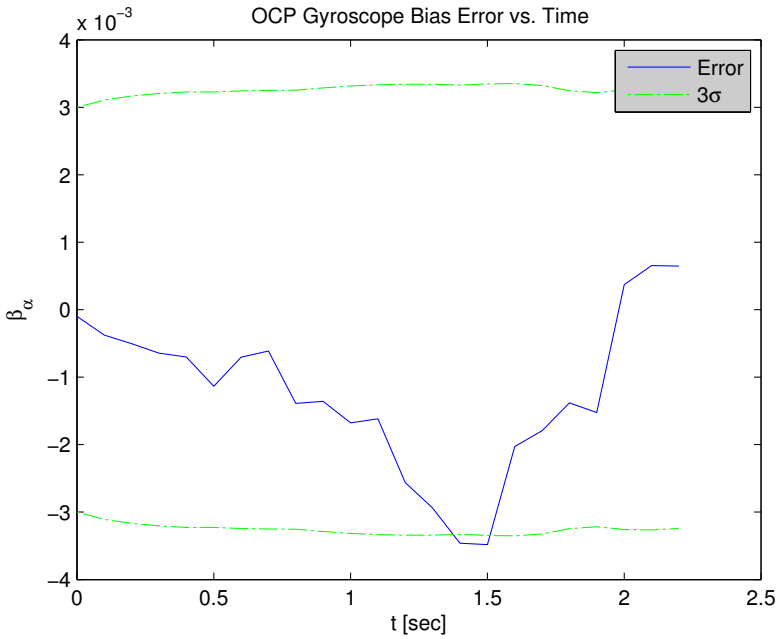


Figure 6.19: Optimal MonoSLAM IMU gyroscopic bias error versus time

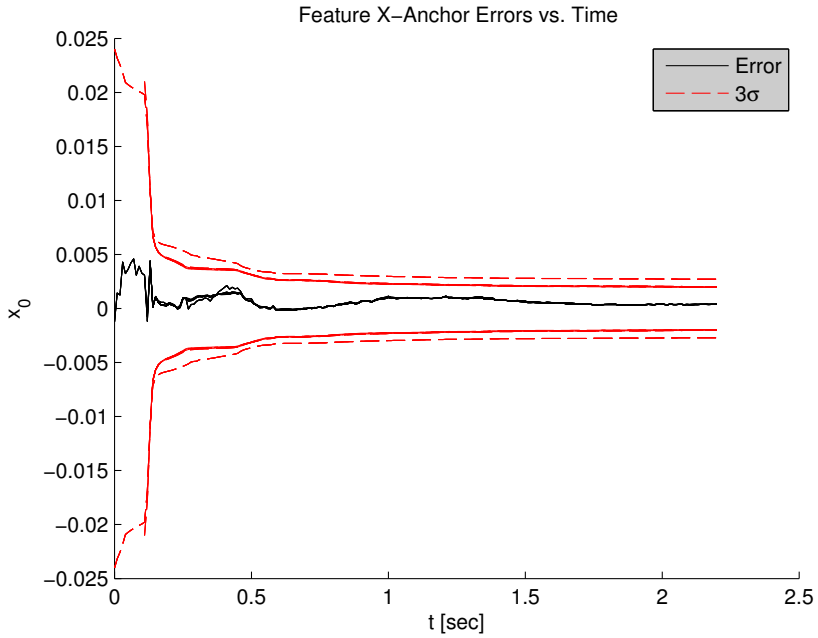


Figure 6.20: Sinusoidal MonoSLAM feature x-anchor errors versus time

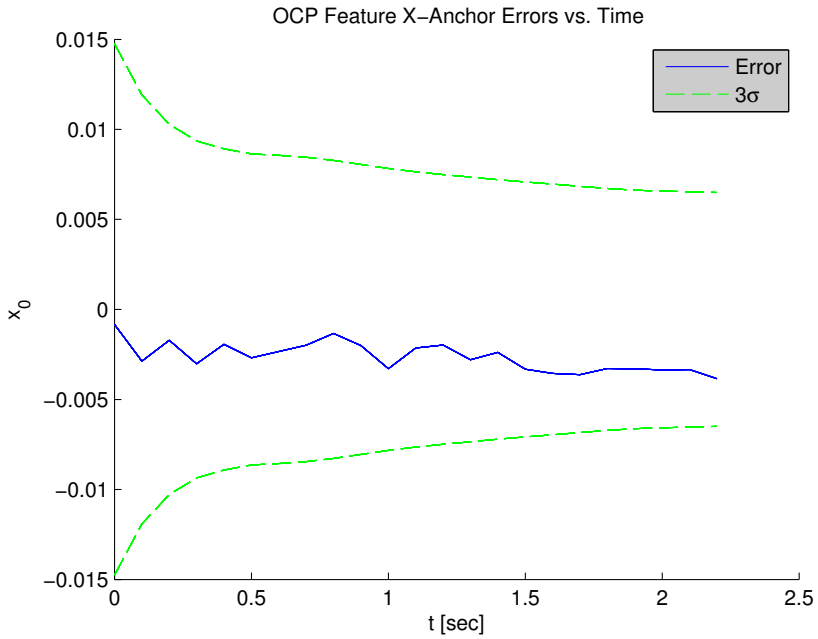


Figure 6.21: Optimal MonoSLAM feature x-anchor errors versus time

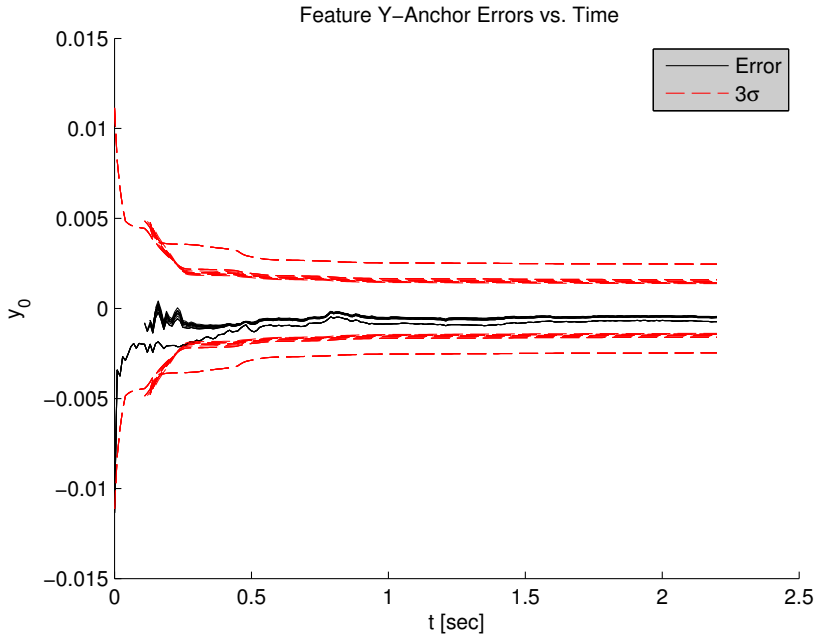


Figure 6.22: Sinusoidal MonoSLAM feature y-anchor errors versus time

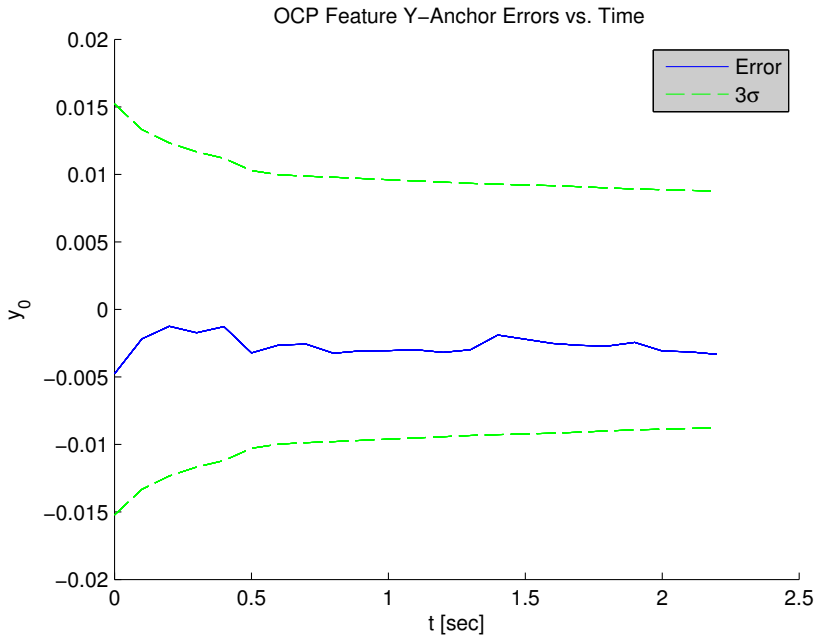


Figure 6.23: Optimal MonoSLAM feature y-anchor errors versus time

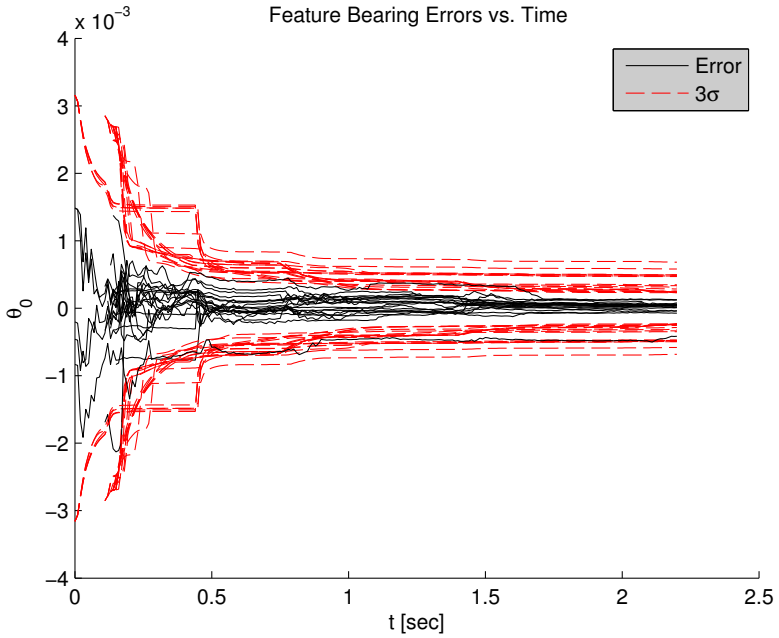


Figure 6.24: Sinusoidal MonoSLAM feature bearing errors versus time

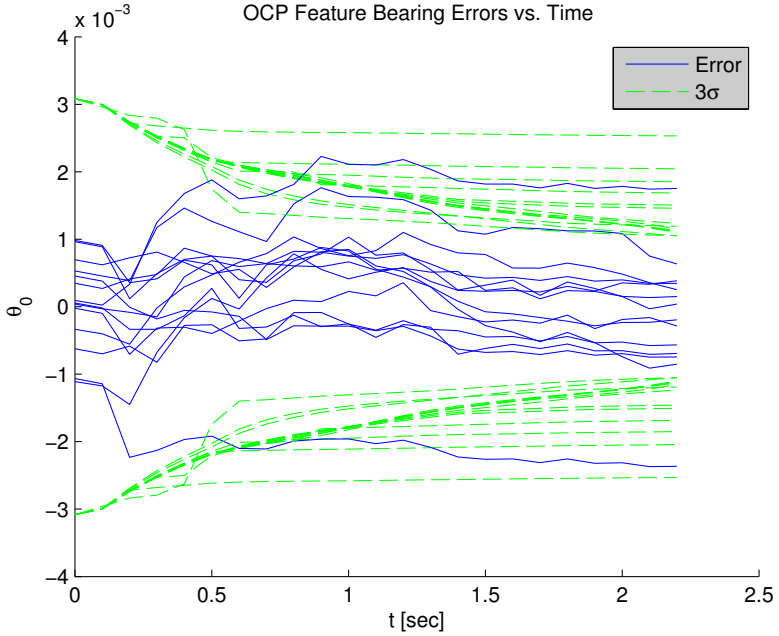


Figure 6.25: Optimal MonoSLAM feature bearing errors versus time

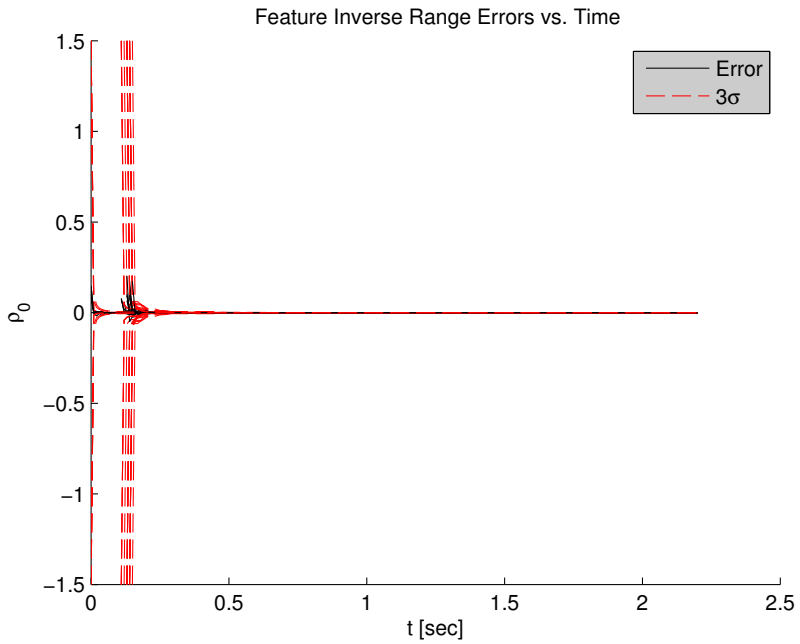


Figure 6.26: Sinusoidal MonoSLAM feature inverse-range errors versus time

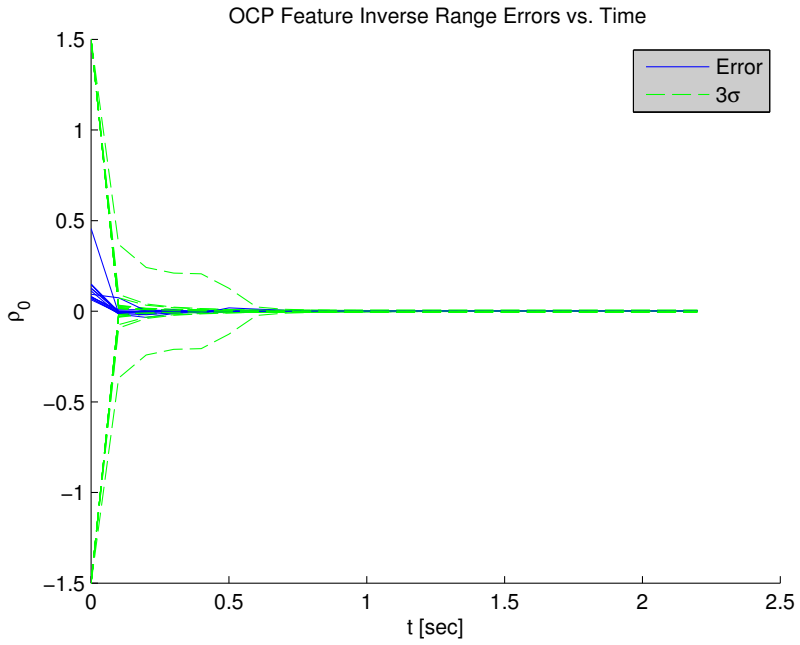


Figure 6.27: Optimal MonoSLAM feature inverse-range errors versus time

From a cursory look at the preceding plots, one can conclude that the sinusoidal path qualitatively provides both better estimation accuracies and tighter covariances than its OPG counterpart. However, it is enlightening to quantify the estimation accuracy of both methods. The author's calculus for creating a value indicative of estimation error is called the error integral:

$$\frac{1}{2}\int_{t_0}^{t_f} \|e\| \, \mathrm{d}t$$

where e is the error of the state in question. For feature state errors, a slight modification is made.

$$\frac{1}{2}\frac{1}{N_{\text{init}}}\sum_{\text{i}=1}^{N_{\text{init}}}\left\{\int_{t_{\text{i}}}^{t_0} \left\|e^{\text{i}}\right\| \, \mathrm{d}t\right\}$$

Here, the integrated feature state error magnitude is averaged over the unknown features that have been seen and initialized. N_{init} is the number of unknown features that were initialized in the simulation, t_{i} is the time step at which the i-th unknown feature was first seen, and e^{i} is the state error in question of the i-th unknown feature. Table 6.2 contains the error integrals of both methods. The numbers in the far right-hand column are the error integrals from the OPG simulation divided by the error integrals from the sinusoid simulation.

Table 6.2: Error integral data

State	OPG Error Integrals	Sinusoid Error Integrals	Units	Ratio
x	0.0022	0.0008	$\mathrm{m} \cdot \mathrm{s}$	2.7906
y	0.0055	0.0006	$\mathrm{m} \cdot \mathrm{s}$	9.7604
v_x	0.0072	0.0036	m	1.9804
v_y	0.0101	0.0033	m	3.0133
α	0.0009	0.0001	$\mathrm{rad} \cdot \mathrm{s}$	7.4812
β_x	0.0073	0.0044	$\frac{\mathrm{m}}{\mathrm{s}}$	1.6756
β_y	0.0113	0.0142	$\frac{\mathrm{m}}{\mathrm{s}}$	0.7975
β_α	0.0016	0.0001	rad	12.9758
$\bar{x}_0$	0.0029	0.0010	$\mathrm{m} \cdot \mathrm{s}$	3.0162
$\bar{y}_0$	0.0029	0.0010	$\mathrm{m} \cdot \mathrm{s}$	2.8079
$\bar{\theta}_0$	0.0008	0.0002	$\mathrm{rad} \cdot \mathrm{s}$	3.4209
$\bar{\rho}_0$	0.0051	0.0008	$\mathrm{m}^{-1} \cdot s$	6.6357

It is evident from the ratio values in Table 6.2 that the sinusoidal path provides better estimation accuracy in this simulation. However, it is also worth investigating the time-averaged error integrals of both methods. This is accomplished by dividing the error integrals by the total simulation time, $t_f = 2.2$ seconds. The resulting values are named the average errors and are given in Table 6.3.

Table 6.3: Time-averaged errors

State	OPG Average Error	Sinusoid Average Error	Units
x	0.0010	0.0004	m
y	0.0025	0.0003	m
v_x	0.0033	0.0016	$\frac{\text{m}}{\text{s}}$
v_y	0.0046	0.0015	$\frac{\text{m}}{\text{s}}$
α	0.0004	0.00005	rad
β_x	0.0033	0.0020	$\frac{\text{m}}{\text{s}^2}$
β_y	0.0051	0.0065	$\frac{\text{m}}{\text{s}^2}$
β_α	0.0007	0.00005	$\frac{\text{rad}}{\text{s}}$
$\bar{x}_0$	0.0013	0.0005	m
$\bar{y}_0$	0.0013	0.0005	m
$\bar{\theta}_0$	0.0004	0.0001	rad
$\bar{\rho}_0$	0.0023	0.0004	m^{-1}

The values in Table 6.3 indicate that even though the sinusoidal path provides better estimation accuracy than the OPG method, the accuracy achieved by the OPG method is adequate for many applications. For instance, the averages of the time-averaged OPG position and velocity errors are approximately 0.2 cm and 0.4 $\frac{\text{cm}}{\text{s}}$, respectively. Accuracy needs will vary from application to application, but the OPG average errors from this simulation are sufficient for a wide variety of uses.

6.3 Monte Carlo Simulation

In order to better examine the behavior of the OPG MonoSLAM method with respect to typical MonoSLAM usages, a Monte Carlo simulation with $M = 2000$

runs was performed. In each run, the vehicle start and finish positions and feature locations (both known and unknown) were randomized. Additionally, the standard deviations of measurement noise and initial heading uncertainty were changed from 0.05° to 0.1°. Otherwise, the simulation performed here 2000 times is the same simulation as the comparison simulation from the previous section.

During each Monte Carlo iteration, the OPG MonoSLAM and the Sinusoidal MonoSLAM algorithms were implemented, producing two sets of estimation data. The fuel and error integrals from Sections 6.1 and 6.2, respectively, were calculated for each iteration, and then averaged over all iterations.

Interestingly, all 2000 sinusoidal path iterations estimated well, whereas there were 22 iterations (approximately 1% of all iterations) in which the OPG method clearly exhibited EKF divergence. EKF divergence happens occasionally in MonoSLAM for several reasons (high noise, feature lying directly on top of the vehicle path, etc.) which all have the effect of increasing the non-linearity in the system. However, in this case it is likely that the IMU readings in the OPG algorithm did not overcome the noise in the IMU (i.e. vehicle accelerations and heading-rates were too low). Tables 6.4 and 6.5 were created by averaging the fuel and error integrals over the iterations in which the OPG method did not exhibit EKF divergence.

Table 6.4: Monte Carlo fuel integral data

	OPG	Sinusoid	Units	Ratio
Fuel Integral	11.1964	106.8167	$\frac{\mathrm{m}}{s}$	0.1048

Table 6.5: Monte Carlo error integral data

State	OPG Error Integrals	Sinusoid Error Integrals	Units	Ratio
x	0.0218	0.0021	$\mathrm{m} \cdot \mathrm{s}$	10.4184
y	0.0211	0.0024	$\mathrm{m} \cdot \mathrm{s}$	8.6331
v_x	0.0423	0.0069	m	6.1500
v_y	0.0459	0.0064	m	7.1625
α	0.0033	0.0003	$\mathrm{rad} \cdot \mathrm{s}$	10.1373
β_x	0.0355	0.0080	$\frac{\mathrm{m}}{\mathrm{s}}$	4.3277
β_y	0.0344	0.0100	$\frac{\mathrm{m}}{\mathrm{s}}$	3.4340
β_α	0.0043	0.0006	rad	7.4470
$\bar{x}_0$	0.0176	0.0028	$\mathrm{m} \cdot \mathrm{s}$	6.2993
$\bar{y}_0$	0.0150	0.0037	$\mathrm{m} \cdot \mathrm{s}$	4.0086
$\bar{\theta}_0$	0.0030	0.0009	$\mathrm{rad} \cdot \mathrm{s}$	3.4752
$\bar{\rho}_0$	0.0390	0.0064	$\mathrm{m}^{-1} \cdot s$	6.0611

Over 2000 iterations, the average fuel used by a vehicle navigating via the OPG algorithm expressed in percentage of the average fuel used by a vehicle navigating via vanilla MonoSLAM was a mere 10.5%. This represents a fuel savings of 89.5%. This is an outstanding improvement upon the typical sinusoid path for MonoSLAM. However, the estimation performance of the OPG is not as good as that of the typical sinusoidal path. From the Monte Carlo data, the OPG error integrals range between $3 - 10$ times the value of the sinusoid error integrals. Additionally, that is only true for runs which did not exhibit EKF divergence; only the OPG algorithm suffered from EKF divergence in a total of 2000 runs, again indicative of the superior

estimation performance the sinusoidal path provides.

It is again enlightening to examine the error data for both methods and not simply the ratio of the errors. Table 6.6 provides data obtained by dividing the error integrals from Table 6.5 by the total simulation time, $t_f = 2.2$ seconds.

Table 6.6: Time-averaged Monte Carlo errors

State	OPG Average Error	Sinusoid Average Error	Units
x	0.0099	0.0010	m
y	0.0096	0.0011	m
v_x	0.0192	0.0031	$\frac{\text{m}}{\text{s}}$
v_y	0.0209	0.0029	$\frac{\text{m}}{\text{s}}$
α	0.0015	0.0001	rad
β_x	0.0161	0.0036	$\frac{\text{m}}{\text{s}^2}$
β_y	0.0156	0.0045	$\frac{\text{m}}{\text{s}^2}$
β_α	0.0020	0.0003	$\frac{\text{rad}}{\text{s}}$
$\bar{x}_0$	0.0080	0.0013	m
$\bar{y}_0$	0.0068	0.0017	m
$\bar{\theta}_0$	0.0014	0.0004	rad
$\bar{\rho}_0$	0.0177	0.0029	m^{-1}

Similar to the errors examined in Section 6.2, it is clear that the values in Table 6.6 indicate (for a large variety of applications) an accuracy that is more than adequate. For instance, the Monte Carlo-averaged position and velocity error integrals are approximately 1 cm and 2 $\frac{\text{cm}}{\text{s}}$, respectively. It is the opinion of the author that this

level of accuracy combined with the overwhelming fuel savings produced by the OPG algorithm indicate that it is the superior method.

7. CONCLUSIONS

Typical MonoSLAM paths include a sinusoidal curve in the $X - Y$ plane, which ensures enough feature parallax and IMU excitation to provide good estimation performance. However, no one has yet to rigorously question if better MonoSLAM paths exist. The optimal path generation method set out in this thesis maximizes feature parallax, and as a result, generally provides enough IMU excitation in the form of accelerations and heading rates in order to create good estimation performance. More significantly, the optimal path generation method uses much less acceleration than the sinusoid paths and creates and more direct path. As a result, fuel savings on the order of 98.8% have been seen. This is extraordinary.

As discussed in Sections 6.2 and 6.3, while there are massive fuel savings with OPG, there is also a loss of estimation accuracy. As revealed in the Monte Carlo simulation, a sinusoidal MonoSLAM path achieved estimation accuracy on the order of 3-10 times better than the OPG method. However, when looking at the estimation errors of both methods separately, it was evident that the OPG paths provided a level of estimation accuracy that is sufficient for many applications.

At the very least, this work has shown that there is room to improve upon the typically-used sinusoidal paths. For instance, since the author has shown paths which provide (on average) approximately 90% fuel savings with 3-10 times poorer estimation accuracy, then perhaps paths exist that provide 50% fuel savings without a loss of estimation accuracy. However, the author's optimal path generation method has shown enough merit in simulation to be worthy of hardware inclusion as is.

7.1 Future Work

One potential issue with this work is computation time. The method formulates an optimal control problem, discretizes it, and then solves it using the fmincon routine in MATLAB. The function solves for vehicle x and y positions and global x and y velocities. For a problem with N time steps, fmincon solves for $4N - 2$ variables; as the number of time steps increases, the number of variables that must be solved for increases four-fold. Additionally, several reformulations of this problem must be solved, although with less states each time as the vehicle comes closer and closer to the desired finish position. At around 25 time steps, this solution process slows down significantly. For simulation purposes, this is not an issue. But, this is an issue when considering hardware implementation and real-world use. However, this does not eliminate the method from hardware inclusion.

One possible solution to this issue is to employ parallel computing. The author did not have access to parallel computing when performing the simulations within this thesis. Additionally, it is worth investigating how the path changes with the number of time steps. If little to no change happens with the path as the number of discrete data points is increased or decreased, then one could simply decrease the number of steps until the computation time is no longer a burden on the hardware.

Additionally, instead of directly implementing the OPG algorithm on hardware, it can be used indirectly. One could use the OPG simulation as a tool to help predict what an optimal path should look like, given an environment, and then implement these path predictions on hardware and compare with other typical sinusoidal paths as well as the optimal path given by simulation. There will be some loss of optimality with this technique; however, the potential gain in simplicity would make this a viable process.

Currently the author's optimal path generation method exists only in a simplified scheme: two dimensional, vehicle heading is fixed along the vehicle path, all features are modeled as points, etc. A more robust simulation (three dimensional, variable heading, more complex features, etc.) is attainable, and would allow for the extension of insight from unmanned land vehicles to unmanned aerial vehicles: quad-rotors, fixed-wing drones, etc.

REFERENCES

[1] Prentice, S. and Roy, N., "The Belief Roadmap: Efcient Planning in Linear POMDPs by Factoring the Covariance," *Robotics Research - The 13th International Symposium ISRR*, Springer, New York, 2010, pp. 293–305.

[2] Makarenko, A. A., Williams, S. B., Bourgault, F., and Durrant-Whyte, H. F., "An Experiment in Integrated Exploration," *IEEE/RSJ International Conference on Intelligent Robots and Systems 2002*, IEEE, New York, 2002, pp. 534–539.

[3] Kim, A. and Eustice, R., "Perception-Driven Navigation: Active Visual SLAM for Robotic Area Coverage," *2013 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, New York, 2013, pp. 3196–3203.

[4] Stachniss, C., Grisetti, G., and Burgard, W., "Information Gain-based Exploration Using Rao-Blackwellized Particle Filters," *Proceedings of Robotics: Science and Systems*, Cambridge, USA, 2005.

[5] Davison, A., Reid, I., Molton, N., and Stasse, O., "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 6, 2007, pp. 1052–1067.

[6] Sol, J., Vidal-Calleja, T., Civera, J., and Montiel, J. M. M., "Impact of Landmark Parametrization on Monocular EKF-SLAM with Points and Lines," *International Journal of Computer Vision*, Vol. 97, No. 3, 2012, pp. 339–368.

[7] Brink, K., Doucette, E., and Miller, M., "Comparing Traditional and Motion Constraint Methods for EKF-Based SLAM," *Proceedings of the ION 2013 Paific PNT Meeting*, ION, Manassas, 2013, pp. 344–351.

[8] Crassidis, J. and Junkins, J. L., *Optimal Estimation of Dynamic Systems*, CRC Press, Boca Raton, 2nd ed., 2012, p. 188.

[9] Civera, J., Davison, A., and Montiel, J. M. M., "Inverse Depth Parametrization for Monocular SLAM," *IEEE Transactions on Robotics*, Vol. 24, No. 5, 2008, pp. 932–945.

[10] Lewis, F. L., Vrabie, D., and Syrmos, V. L., *Optimal Control*, Wiley, Hoboken, 3rd ed., 2012.